Documentação do Astra Trident 24,02

Astra Trident

NetApp
December 03, 2024

Índice

Documentação do Astra Trident 24,02

Notas de lançamento

O que há de novo

As Notas de versão fornecem informações sobre novos recursos, aprimoramentos e correções de bugs na versão mais recente do Astra Trident.

> O `tridentctl` binário para Linux que é fornecido no arquivo zip do instalador é a versão testada e suportada. Esteja ciente de que o `macos` binário fornecido na `/extras` parte do arquivo zip não é testado ou suportado.

Novidades em 24,02

Melhorias

- Adicionado suporte para o Cloud Identity.
  - AKS com ANF - o Azure Workload Identity será usado como identidade de nuvem.
  - O EKS com FSxN - função do AWS IAM será usado como identidade na nuvem.
- Adicionado suporte para instalar o Astra Trident como um complemento no cluster EKS a partir do console EKS.
- Adicionada capacidade de configurar e desativar a recuperação automática iSCSI ("Problema nº 864").
- A personalidade do FSX foi adicionada aos drivers do ONTAP para permitir a integração com o AWS IAM e o SecretsManager e permitir que o Astra Trident exclua volumes do FSX com backups ("Problema nº 453").

Kubernetes

- Adicionado suporte para Kubernetes 1,29.

Correções

- Mensagens de aviso do ACP fixas, quando o ACP não está ativado ("Problema nº 866").
- Adicionado um atraso de 10 segundos antes de executar uma divisão de clones durante a exclusão de snapshot para drivers ONTAP, quando um clone está associado ao snapshot.

Desvalorizações

- Estrutura de atestações in-toto removida dos manifestos de imagem multi-plataforma.

Mudanças em 23,10

Correções

- Expansão de volume fixa se um novo tamanho solicitado for menor do que o tamanho total do volume para os drivers de armazenamento ONTAP-nas e ONTAP-nas-FlexGroup ("Problema nº 834").
- Tamanho de volume fixo para exibir somente o tamanho utilizável do volume durante a importação para drivers de armazenamento ONTAP-nas e ONTAP-nas-FlexGroup ("Problema nº 722").

- Conversão de nomes FlexVol fixos para ONTAP-nas-Economy.
- Corrigido problema de inicialização do Astra Trident em um nó do Windows quando o nó é reinicializado.

Melhorias

Kubernetes

Adicionado suporte para Kubernetes 1,28.

Astra Trident

- Adicionado suporte para o uso de identidades gerenciadas do Azure (AMI) com o driver de armazenamento azure-NetApp-Files.
- Adicionado suporte para NVMe sobre TCP para o driver ONTAP-SAN.
- Adicionada capacidade de pausar o provisionamento de um volume quando o back-end é definido como estado suspenso pelo usuário ("Problema nº 558").

Recursos avançados disponíveis no Astra Control

Com o Astra Trident 23,10, um novo componente de software chamado Astra Control Provisioner está disponível para usuários licenciados do Astra Control. Esse provisionador fornece acesso a um superconjunto de recursos de provisionamento de storage e gerenciamento avançados além daqueles que o Astra Trident oferece suporte por conta própria. Para a versão 23,10, esses recursos incluem:

- Recursos de backup e restauração para aplicativos com backends de armazenamento com driver ONTAP-nas-Economy
- Segurança de back-end de armazenamento aprimorada com criptografia Kerberos 5
- Recuperação de dados usando um snapshot
- Melhorias no SnapMirror

"Saiba mais sobre o Astra Control Provisioner."

Mudanças em 23.07.1

Kubernetes: exclusão do daemonset fixa para oferecer suporte a atualizações sem inatividade ("Problema nº 740").

Mudanças em 23,07

Correções

Kubernetes

- Atualização do Trident corrigida para ignorar pods antigos presos no estado de terminação ("Problema nº 740").
- Adicionado tolerância à definição "transient-Trident-version-pod" ("Problema nº 795").

Astra Trident

- Solicitações ZAPI ONTAP fixas para garantir que os números de série LUN sejam consultados ao obter atributos de LUN para identificar e corrigir dispositivos iSCSI fantasma durante as operações de

estadiamento de nós.

- Corrigido o erro de manipulação no código do driver de armazenamento ("Problema nº 816").
- Ajuste o tamanho da cota ao usar drivers ONTAP com o uso-REST.
- Criação de clone de LUN fixo em ONTAP-san-Economy.
- Reverter campo de informações de publicação `rawDevicePath` de para `devicePath`; lógica adicionada para preencher e recuperar (em alguns casos) `devicePath` campo.

Melhorias

Kubernetes

- Adicionado suporte para importar instantâneos pré-provisionados.
- Implementação minimizada e permissões do daemonset linux ("Problema nº 817").

Astra Trident

- Não é mais relatar o campo de estado para volumes e instantâneos "online".
- Atualiza o estado de back-end se o back-end do ONTAP estiver off-line ("Problemas nº 801", "Nº 543").
- O número de série LUN é sempre recuperado e publicado durante o fluxo de trabalho ControllerVolumePublish.
- Adicionada lógica adicional para verificar o número de série e o tamanho do dispositivo multipath iSCSI.
- Verificação adicional para volumes iSCSI para garantir que o dispositivo multipath correto seja desorganizado.

Aperfeiçoamento experimental

Adicionado suporte de visualização técnica para NVMe sobre TCP para o driver ONTAP-SAN.

Documentação

Muitas melhorias organizacionais e de formatação foram feitas.

Desvalorizações

Kubernetes

- Suporte removido para instantâneos v1beta1.
- Suporte removido para volumes pré-CSI e classes de armazenamento.
- Mínimo atualizado com suporte de Kubernetes para 1,22.

Mudanças em 23,04

> Forçar a desagregação de volume para volumes ONTAP-SAN-* é compatível apenas com versões Kubernetes com o recurso desativação de nó não-gracioso ativado. Forçar a desligação deve ser ativada no momento da instalação utilizando o `--enable-force-detach` sinalizador do instalador do Trident.

Correções

- Operador Trident fixo para usar localhost IPv6 para instalação quando especificado na especificação.
- Permissões de função de cluster do operador do Trident fixas para serem sincronizadas com as permissões do pacote ("Problema nº 799").
- Corrigido o problema com a inclusão de volume de bloco bruto em vários nós no modo RWX.
- Suporte fixo à clonagem de FlexGroup e importação de volume para volumes SMB.
- Corrigido o problema em que o controlador Trident não podia desligar imediatamente ("Problema nº 811").
- Correção adicionada para listar todos os nomes do grupo igrop associados a um LUN especificado provisionado com drivers ONTAP-San-*.
- Adicionada uma correção para permitir que processos externos sejam executados até a conclusão.
- Corrigido erro de compilação para a arquitetura s390 ("Problema nº 537").
- Corrigido o nível de registo incorreto durante as operações de montagem de volume ("Problema nº 781").
- Corrigido erro de afirmação de tipo potencial ("Problema nº 802").

Melhorias

- Kubernetes:
  - Adicionado suporte para Kubernetes 1,27.
  - Adicionado suporte para importar volumes LUKS.
  - Adicionado suporte para o modo de acesso ao PVC ReadWriteOncePod.
  - Adicionado suporte para Force Detach para volumes ONTAP-SAN-* durante cenários de encerramento de nó não gracioso.
  - Todos os volumes ONTAP-SAN-* agora usarão grupos por nó. Os LUNs só serão mapeados para os grupos enquanto forem publicados ativamente nesses nós para melhorar a nossa postura de segurança. Os volumes existentes serão oportunisticamente comutados para o novo esquema de grupos quando o Trident determinar que é seguro fazê-lo sem afetar cargas de trabalho ativas ("Problema nº 758").
  - Melhor segurança do Trident ao limpar grupos não utilizados gerenciados pelo Trident dos backends ONTAP-SAN-*.
- Adicionado suporte para volumes SMB com o Amazon FSX para os drivers de armazenamento ONTAP-nas-Economy e ONTAP-nas-FlexGroup.
- Adicionado suporte para compartilhamentos SMB com os drivers de storage ONTAP-nas, ONTAP-nas-Economy e ONTAP-nas-FlexGroup.
- Adicionado suporte para arm64 nós ("Problema nº 732").
- Procedimento de encerramento aprimorado do Trident desativando primeiro os servidores API ("Problema nº 811").
- Adicionado suporte de compilação entre plataformas para Windows e hosts arm64 para Makefile; veja BUILD.md.

Desvalorizações

Kubernetes: Os grupos com escopo de back-end não serão mais criados ao configurar drivers ONTAP-san e ONTAP-san-Economy ("Problema nº 758").

Mudanças em 23.01.1

Correções

- Operador Trident fixo para usar localhost IPv6 para instalação quando especificado na especificação.
- Permissões fixas da função de cluster do operador do Trident para estar em sincronia com as permissões do pacote "Problema nº 799".
- Adicionada uma correção para permitir que processos externos sejam executados até a conclusão.
- Corrigido o problema com a inclusão de volume de bloco bruto em vários nós no modo RWX.
- Suporte fixo à clonagem de FlexGroup e importação de volume para volumes SMB.

Mudanças em 23,01

O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Astra Trident antes de atualizar o Kubernetes.

Correções

- Kubernetes: Adicionadas opções para excluir a criação da Diretiva de Segurança do Pod para corrigir instalações do Trident via Helm ("Problemas nº 783, nº 794").

Melhorias

Kubernetes

- Adicionado suporte para Kubernetes 1,26.
- Utilização geral aprimorada de recursos RBAC do Trident ("Problema nº 757").
- Automação adicionada para detetar e corrigir sessões iSCSI quebradas ou obsoletas em nós de host.
- Adicionado suporte para expandir volumes criptografados LUKS.
- Kubernetes: Suporte à rotação de credenciais adicionado para volumes criptografados LUKS.

Astra Trident

- Adicionado suporte para volumes SMB com o Amazon FSX for ONTAP para o driver de armazenamento ONTAP-nas.
- Adicionado suporte para permissões NTFS ao usar volumes SMB.
- Adicionado suporte a pools de storage para volumes do GCP com nível de serviço CVS.
- Adicionado suporte para uso opcional do flexgroupAggregateList ao criar FlexGroups com o driver de armazenamento ONTAP-nas-FlexGroup.
- Desempenho aprimorado para o driver de storage econômico ONTAP nas ao gerenciar vários FlexVols.
- Atualizações de dataLIF habilitadas para todos os drivers de storage nas do ONTAP.
- Atualização da convenção de nomenclatura Trident Deployment e DaemonSet para refletir o sistema operacional do nó host.

Desvalorizações

- Kubernetes: Mínimo atualizado com suporte de Kubernetes para 1,21.
- Os LIFs de dados não devem mais ser especificados ao configurar `ontap-san` ou `ontap-san-economy`

drivers.

Mudanças em 22,10

Você deve ler as seguintes informações críticas antes de atualizar para o Astra Trident 22,10.

** informações críticas sobre o Astra Trident 22.10 **

- O Kubernetes 1,25 agora é compatível com o Trident. É necessário atualizar o Astra Trident para 22,10 antes da atualização para o Kubernetes 1,25.
- O Astra Trident agora reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Correções

- Corrigido um problema específico para o back-end do ONTAP criado usando `credentials` campo que não aparece on-line durante a atualização do 22.07.0 ("Problema nº 759").
- **Docker:** corrigiu um problema que fazia com que o plugin de volume do Docker não iniciasse em alguns ambientes ("Problema nº 548" e "Problema nº 760").
- Corrigido problema de SLM específico para backends de SAN ONTAP para garantir que apenas um subconjunto de LIFs de dados pertencentes a nós de relatório seja publicado.
- Corrigido problema de desempenho em que verificações desnecessárias para iSCSI LUNs aconteceram ao anexar um volume.
- Novas tentativas granulares removidas dentro do fluxo de trabalho iSCSI Astra Trident para falhar rapidamente e reduzir os intervalos de tentativas externas.
- Corrigido o problema em que um erro foi retornado ao lavar um dispositivo iSCSI quando o dispositivo multipath correspondente já estava lavado.

Melhorias

- Kubernetes:
  - Adicionado suporte para Kubernetes 1,25. É necessário atualizar o Astra Trident para 22,10 antes da atualização para o Kubernetes 1,25.
  - Adicionado um ServiceAccount separado, ClusterRole e ClusterRoleBinding para a implantação do Trident e DaemonSet para permitir melhorias futuras de permissões.
  - Adicionado suporte para "compartilhamento de volume entre namespace".
- Todos os drivers de storage Trident `ontap-*` agora funcionam com a API REST do ONTAP.
- Adicionado novo operador yaml (`bundle_post_1_25.yaml`) sem um `PodSecurityPolicy` para oferecer suporte ao Kubernetes 1,25.
- Adicionado "Suporte para volumes criptografados com LUKS" para `ontap-san` e `ontap-san-economy` drivers de armazenamento.
- Adicionado suporte para nós do Windows Server 2019.

- Adicionado "Suporte para volumes SMB em nós do Windows" através do `azure-netapp-files` driver de armazenamento.
- A deteção automática de comutação MetroCluster para controladores ONTAP está agora disponível em geral.

Desvalorizações

- **Kubernetes:** atualizado com o mínimo de Kubernetes compatível para 1,20.
- Driver do Astra Data Store (ADS) removido.
- Removido o suporte `yes` e `smart` as opções para `find_multipaths` quando configurar multipathing de nó de trabalho para iSCSI.

Mudanças em 22,07

Correções

Kubernetes

- Corrigido problema para lidar com valores booleanos e numéricos para o seletor de nó ao configurar o Trident com Helm ou o Operador Trident. ("GitHub Edição nº 700")
- Corrigido problema no tratamento de erros do caminho não-CHAP, de modo que kubelet irá tentar novamente se falhar. "GitHub Edição nº 736")

Melhorias

- Transição do k8s.gcr.io para o registry.k8s.io como Registro padrão para imagens CSI
- Os volumes ONTAP-SAN agora usarão grupos por nó e mapearão apenas LUNs para grupos enquanto são publicados ativamente nesses nós para melhorar nossa postura de segurança. Os volumes existentes serão oportunisticamente comutados para o novo esquema do grupo quando o Astra Trident determinar que é seguro fazê-lo sem afetar cargas de trabalho ativas.
- Incluído um ResourceQuota com instalações Trident para garantir que o Trident DaemonSet seja programado quando o consumo de PriorityClass é limitado por padrão.
- Adicionado suporte para recursos de rede ao driver Azure NetApp Files. ("GitHub Edição nº 717")
- Adicionada deteção automática de comutação MetroCluster de pré-visualização técnica aos drivers ONTAP. ("GitHub Edição nº 228")

Desvalorizações

- **Kubernetes:** atualizado com o mínimo de Kubernetes compatível para 1,19.
- A configuração de backend não permite mais vários tipos de autenticação em uma única configuração.

Remoções

- O driver do AWS CVS (obsoleto desde 22,04) foi removido.
- Kubernetes
  - Removido recurso SYS_ADMIN desnecessário dos pods de nós.
  - Reduz o nodeprep para informações simples de host e descoberta de serviço ativo para confirmar o melhor esforço de que os serviços NFS/iSCSI estão disponíveis nos nós de trabalho.

Documentação

Uma nova "Padrões de segurança do pod"seção (PSS) foi adicionada detalhando as permissões habilitadas pelo Astra Trident na instalação.

Mudanças em 22,04

A NetApp está continuamente melhorando e aprimorando seus produtos e serviços. Aqui estão alguns dos recursos mais recentes do Astra Trident. Para versões anteriores, "Versões anteriores da documentação" consulte .

> Se você estiver atualizando de qualquer versão anterior do Trident e usar o Azure NetApp Files, o `location` parâmetro config agora é um campo único obrigatório.

Correções

- Análise melhorada de nomes de iniciadores iSCSI. ("GitHub Edição nº 681")
- Corrigido problema em que os parâmetros da classe de armazenamento CSI não eram permitidos. ("GitHub Edição nº 598")
- Declaração de chave duplicada corrigida no CRD Trident. ("GitHub Edição nº 671")
- Corrigidos registos de instantâneos do CSI imprecisos. ("GitHub Edição nº 629"))
- Corrigido o problema com a remoção de volumes em nós excluídos. ("GitHub Edição nº 691")
- Adição de manipulação de inconsistências de sistema de arquivos em dispositivos de bloco. ("GitHub Edição nº 656")
- Corrigido problema ao puxar imagens de suporte automático ao definir o `imageRegistry` sinalizador durante a instalação. ("GitHub Edição nº 715")
- Corrigido o problema em que o driver Azure NetApp Files não conseguiu clonar um volume com várias regras de exportação.

Melhorias

- As conexões de entrada para os endpoints seguros da Trident agora exigem um mínimo de TLS 1,3. ("GitHub Edição nº 698")
- O Trident agora adiciona cabeçalhos HSTS às respostas de seus endpoints seguros.
- O Trident agora tenta ativar o recurso de permissões unix do Azure NetApp Files automaticamente.
- **Kubernetes**: O daemonset do Trident agora é executado na classe de prioridade crítica do nó do sistema. ("GitHub Edição nº 694")

Remoções

O driver da série e (desativado desde 20,07) foi removido.

Mudanças em 22.01.1

Correções

- Corrigido o problema com a remoção de volumes em nós excluídos. ("GitHub Edição nº 691")
- Corrigido o pânico ao acessar campos nil para espaço agregado nas respostas da API do ONTAP.

Mudanças em 22.01.0

Correções

- **Kubernetes:** aumente o tempo de repetição do backoff do Registro de nós para clusters grandes.
- Corrigido problema em que o driver azure-NetApp-Files poderia ser confundido por vários recursos com o mesmo nome.
- Os LIFs de dados SAN IPv6 da ONTAP agora funcionam se especificados com colchetes.
- Corrigido o problema em que a tentativa de importar um volume já importado retorna EOF deixando PVC em estado pendente. ("GitHub Edição nº 489")
- Corrigido o problema quando a performance do Astra Trident diminui quando são criados snapshots > 32 em um volume SolidFire.
- Substituído SHA-1 por SHA-256 na criação de certificado SSL.
- Driver Azure NetApp Files fixo para permitir nomes de recursos duplicados e limitar as operações a um único local.
- Driver Azure NetApp Files fixo para permitir nomes de recursos duplicados e limitar as operações a um único local.

Melhorias

- Melhorias do Kubernetes:
  - Adicionado suporte para Kubernetes 1,23.
  - Adicione opções de agendamento para pods Trident quando instalado via Operador Trident ou Helm. ("GitHub Edição nº 651")
- Permitir volumes entre regiões no driver do GCP. ("GitHub Edição nº 633")
- Adicionado suporte para a opção 'unixPermissions' para volumes Azure NetApp Files. ("GitHub Edição nº 666")

Desvalorizações

A interface REST do Trident pode ouvir e servir apenas em endereços 127.0.0.1 ou [::1]

Mudanças em 21.10.1

A versão v21.10.0 tem um problema que pode colocar o controlador Trident em um estado CrashLoopBackOff quando um nó é removido e depois adicionado de volta ao cluster do Kubernetes. Esse problema foi corrigido no v21,10.1 (GitHub Issue 669).

Correções

- Condição de corrida potencial fixa ao importar um volume em um back-end CVS do GCP, resultando em falha na importação.
- Corrigido um problema que pode colocar o controlador Trident em um estado CrashLoopBackOff quando um nó é removido e depois adicionado de volta ao cluster do Kubernetes (problema 669 do GitHub).
- Corrigido o problema em que os SVMs não eram mais descobertos se nenhum nome SVM foi especificado (problema 612 do GitHub).

Mudanças em 21.10.0

Correções

- Corrigido o problema em que clones de volumes XFS não podiam ser montados no mesmo nó que o volume de origem (problema 514 do GitHub).
- Corrigido o problema em que o Astra Trident registrou um erro fatal no desligamento (problema 597 do GitHub).
- Correções relacionadas ao Kubernetes:
  - Retorne o espaço usado de um volume como o mínimo restoresSize ao criar snapshots com `ontap-nas` drivers e `ontap-nas-flexgroup` (GitHub Issue 645).
  - Corrigido o problema em que `Failed to expand filesystem` o erro foi registrado após o redimensionamento de volume (GitHub problema 560).
  - Corrigido o problema em que um pod poderia ficar preso `Terminating` no estado (GitHub problema 572).
  - Corrigido o caso em que um `ontap-san-economy` FlexVol pode estar cheio de LUNs instantâneos (GitHub problema 533).
  - Corrigido o problema do instalador personalizado YAML com imagem diferente (problema 613 do GitHub).
  - Corrigido cálculo do tamanho do instantâneo (GitHub edição 611).
  - Corrigido o problema em que todos os instaladores do Astra Trident podiam identificar o Kubernetes simples como OpenShift (problema 639 do GitHub).
  - Corrigido o operador do Trident para parar a reconciliação se o servidor da API do Kubernetes não estiver acessível (problema 599 do GitHub).

Melhorias

- Adicionado suporte à `unixPermissions` opção para volumes de performance do GCP-CVS.
- Adicionado suporte para volumes CVS otimizados para escala no GCP na faixa de 600 GiB a 1 TIB.
- Aprimoramentos relacionados ao Kubernetes:
  - Adicionado suporte para Kubernetes 1,22.
  - Habilitou o operador do Trident e o gráfico Helm para trabalhar com o Kubernetes 1,22 (GitHub Issue 628).
  - Adicionado a imagem do operador ao `tridentctl` comando imagens (GitHub Issue 570).

Melhorias experimentais

- Adicionado suporte para replicação de volume no `ontap-san` driver.
- Adicionado suporte REST **Tech Preview** para os `ontap-nas-flexgroup` drivers , `ontap-san`, e `ontap-nas-economy` .

Problemas conhecidos

Problemas conhecidos identificam problemas que podem impedi-lo de usar o produto com sucesso.

- Ao atualizar um cluster do Kubernetes do 1,24 para o 1,25 ou posterior que tenha o Astra Trident instalado, você deve atualizar o Values.yaml para definir `excludePodSecurityPolicy` `true` ou adicionar `--set excludePodSecurityPolicy=true` `helm upgrade` ao comando antes de atualizar o cluster.
- Agora, o Astra Trident aplica um espaço em `fsType` (`fsType=""` `branco ) para volumes que não têm o ` `fsType` especificado em seu StorageClass. Ao trabalhar com o Kubernetes 1,17 ou posterior, a Trident dá suporte a fornecer um espaço em branco `fsType` para volumes NFS. Para volumes iSCSI, é necessário definir o `fsType` no StorageClass ao aplicar um `fsGroup` contexto de uso de segurança.
- Ao usar um back-end em várias instâncias do Astra Trident, cada arquivo de configuração de back-end deve ter um valor diferente `storagePrefix` para backends do ONTAP ou usar um diferente `TenantName` para backends do SolidFire. O Astra Trident não consegue detectar volumes que outras instâncias do Astra Trident criaram. Tentar criar um volume existente em backends ONTAP ou SolidFire é bem-sucedido, porque o Astra Trident trata a criação de volume como uma operação idempotente. Se `storagePrefix` ou `TenantName` não forem diferentes, pode haver colisões de nomes para volumes criados no mesmo back-end.
- Ao instalar o Astra Trident (usando `tridentctl` ou o Operador Trident) e usar `tridentctl` para gerenciar o Astra Trident, você deve garantir que a `KUBECONFIG` variável de ambiente esteja definida. Isso é necessário para indicar o cluster do Kubernetes com `tridentctl` quem trabalhar. Ao trabalhar com vários ambientes do Kubernetes, você deve garantir que o `KUBECONFIG` arquivo seja obtido com precisão.
- Para executar a recuperação de espaço on-line para PVS iSCSI, o SO subjacente no nó de trabalho pode exigir que as opções de montagem sejam passadas para o volume. Isso é verdade para instâncias RHEL/RedHat CoreOS, que exigem o `discard` "opção de montagem"; Certifique-se de que a opção Descartar mountOption está incluída no seu[`StorageClass`site para suportar descarte de blocos online.
- Se você tiver mais de uma instância do Astra Trident por cluster Kubernetes, o Astra Trident não poderá se comunicar com outras instâncias e não poderá descobrir outros volumes que eles criaram, o que leva a um comportamento inesperado e incorreto se mais de uma instância for executada em um cluster. Só deve haver uma instância do Astra Trident por cluster Kubernetes.
- Se os objetos baseados no Astra Trident `StorageClass` forem excluídos do Kubernetes enquanto o Astra Trident estiver off-line, o Astra Trident não removerá as classes de storage correspondentes de seu banco de dados quando ele voltar on-line. Você deve excluir essas classes de armazenamento usando `tridentctl` ou a API REST.
- Se um usuário excluir um PV provisionado pelo Astra Trident antes de excluir o PVC correspondente, o Astra Trident não excluirá automaticamente o volume de backup. Você deve remover o volume via `tridentctl` ou a API REST.
- A ONTAP não pode provisionar simultaneamente mais de um FlexGroup de cada vez, a menos que o conjunto de agregados seja exclusivo para cada solicitação de provisionamento.
- Ao usar o Astra Trident mais de IPv6 TB, você deve especificar `managementLIF` e `dataLIF` na definição de back-end entre colchetes. Por exemplo, `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`.

> Não é possível especificar `dataLIF` em um back-end de SAN ONTAP. O Astra Trident descobre todas as LIFs iSCSI disponíveis e as usa para estabelecer a sessão multipath.

- Se estiver usando `solidfire-san` o driver com OpenShift 4,5, certifique-se de que os nós de trabalho subjacentes usem MD5 como o algoritmo de autenticação CHAP. Os algoritmos CHAP seguros compatíveis com FIPS SHA1, SHA-256 e SHA3-256 estão disponíveis com o Element 12,7.

Encontre mais informações

- "Astra Trident no GitHub"
- "Blogs do Astra Trident"

Versões anteriores da documentação

Se você não estiver executando o Astra Trident 24,02, a documentação das versões anteriores estará disponível com base "Ciclo de vida do suporte ao Astra Trident"no .

- "Astra Trident 23,10"
- "Astra Trident 23,07"
- "Astra Trident 23,04"
- "Astra Trident 23,01"
- "Astra Trident 22,10"
- "Astra Trident 22,07"
- "Astra Trident 22,04"
- "Astra Trident 22,01"

Comece agora

Saiba mais sobre o Astra Trident

Saiba mais sobre o Astra Trident

O Astra Trident é um projeto de código aberto com suporte total mantido pela NetApp como parte "Família de produtos Astra"do . Ele foi desenvolvido para ajudar você a atender às demandas de persistência da sua aplicação em contêineres usando interfaces padrão do setor, como a Container Storage Interface (CSI).

O que é o Astra?

Com o Astra, é mais fácil para as empresas gerenciar, proteger e mover workloads em contêineres com muitos dados executados no Kubernetes dentro e entre nuvens públicas e no local.

O Astra provisiona e fornece storage de contêiner persistente com base no Astra Trident. Ele também oferece recursos avançados de gerenciamento de dados com reconhecimento de aplicações, como snapshot, backup e restauração, logs de atividade e clonagem ativa para proteção de dados, recuperação de desastres/dados, auditoria de dados e casos de uso de migração para workloads Kubernetes.

Saiba mais "Inscreva-se para uma avaliação gratuita"sobre o .

O que é o Astra Trident?

O Astra Trident permite o consumo e gerenciamento de recursos de storage em todas as plataformas de storage populares da NetApp, na nuvem pública ou no local, incluindo ONTAP (AFF, FAS, Select, nuvem, Amazon FSX for NetApp ONTAP), software Element (NetApp HCI, SolidFire), serviço Azure NetApp Files e Cloud Volumes Service no Google Cloud.

O Astra Trident é um orquestrador de storage dinâmico e em conformidade com "Kubernetes"a Container Storage Interface (CSI) que se integra nativamente ao . O Astra Trident é executado como um único Pod de controladora e um pod de nó em cada nó de trabalho no cluster. "A arquitetura do Astra Trident"Consulte para obter detalhes.

O Astra Trident também fornece integração direta com o ecossistema Docker para plataformas de storage NetApp. O plug-in de volume do Docker do NetApp (nDVP) dá suporte ao provisionamento e gerenciamento de recursos de storage da plataforma de storage para hosts do Docker. "Implante o Astra Trident para Docker"Consulte para obter detalhes.

> Se esta for a primeira vez que usa o Kubernetes, você deve se familiarizar com o "Conceitos e ferramentas do Kubernetes".

Faça o test drive do Astra Trident

Para fazer um test drive, solicite acesso ao "Deploy and Clone Persistent Storage for containerized workloads" "Unidade de teste do NetApp" usando uma imagem de laboratório pronta para uso. A unidade de teste fornece um ambiente sandbox com um cluster de Kubernetes de três nós e o Astra Trident instalado e configurado. Esta é uma ótima maneira de se familiarizar com o Astra Trident e explorar seus recursos.

Outra opção é a "Guia de Instalação do kubeadm" fornecida pelo Kubernetes.

Não use um cluster do Kubernetes criado usando essas instruções em um ambiente de produção. Use os guias de implantação de produção fornecidos pela distribuição para clusters prontos para produção.

Integração do Kubernetes com os produtos NetApp

O portfólio de produtos de storage do NetApp se integra a muitos aspectos de um cluster Kubernetes, fornecendo recursos avançados de gerenciamento de dados que melhoram o recurso, a funcionalidade, a performance e a disponibilidade da implantação do Kubernetes.

Amazon FSX para NetApp ONTAP

"Amazon FSX para NetApp ONTAP" É um serviço AWS totalmente gerenciado que permite iniciar e executar sistemas de arquivos equipados com o sistema operacional de storage NetApp ONTAP.

Azure NetApp Files

"Azure NetApp Files" É um serviço de compartilhamento de arquivos do Azure de nível empresarial, desenvolvido pela NetApp. É possível executar os workloads mais exigentes baseados em arquivos no Azure de forma nativa, com a performance e o gerenciamento de rich data que você espera do NetApp.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" É um dispositivo de storage somente de software que executa o software de gerenciamento de dados ONTAP na nuvem.

Cloud Volumes Service para Google Cloud

"NetApp Cloud Volumes Service para Google Cloud" É um serviço de arquivos nativo da nuvem que fornece volumes nas em NFS e SMB com performance all-flash.

Software Element

"Elemento" permite que o administrador de storage consolide workloads garantindo a performance e possibilitando um espaço físico do storage simplificado e otimizado.

NetApp HCI

"NetApp HCI" simplifica o gerenciamento e a escala do data center automatizando tarefas de rotina e permitindo que os administradores de infraestrutura se concentrem em funções mais importantes.

O Astra Trident pode provisionar e gerenciar dispositivos de storage para aplicações em contêiner diretamente na plataforma de storage subjacente da NetApp HCI.

NetApp ONTAP

"NetApp ONTAP" É o sistema operacional de storage unificado multiprotocolo da NetApp que oferece recursos avançados de gerenciamento de dados para qualquer aplicação.

Os sistemas ONTAP têm configurações all-flash, híbridas ou totalmente HDD e oferecem muitos modelos de implantação diferentes, incluindo hardware projetado (FAS e AFF), white-box (ONTAP Select) e somente para nuvem (Cloud Volumes ONTAP). O Astra Trident é compatível com esses modelos de implantação do ONTAP.

Para mais informações

- "Família de produtos NetApp Astra"
- "Documentação do Astra Control Service"
- "Documentação do Astra Control Center"
- "Documentação da API Astra"

A arquitetura do Astra Trident

O Astra Trident é executado como um único Pod de controladora e um pod de nó em cada nó de trabalho no cluster. O pod de nó deve estar em execução em qualquer host onde você queira potencialmente montar um volume Astra Trident.

Compreensão dos pods dos nós e dos pods do controlador

O Astra Trident é implantado como Pod do controlador Tridentum único e um ou mais Pods do nó Trident no cluster Kubernetes e usa contentores padrão do Kubernetes *CSI Sidecar* para simplificar a implantação de plug-ins do CSI. "Kubernetes CSI Sidecar contêineres" São mantidos pela comunidade do Kubernetes Storage.

Kubernetes "seletores de nós" e "tolerações e taints"são usados para restringir um pod a ser executado em um nó específico ou preferencial. Você pode configurar seletores de nós e tolerâncias para pods de nó e controlador durante a instalação do Astra Trident.

- O plugin controlador lida com o provisionamento e gerenciamento de volume, como snapshots e redimensionamento.
- O plug-in do nó manipula a conexão do armazenamento ao nó.

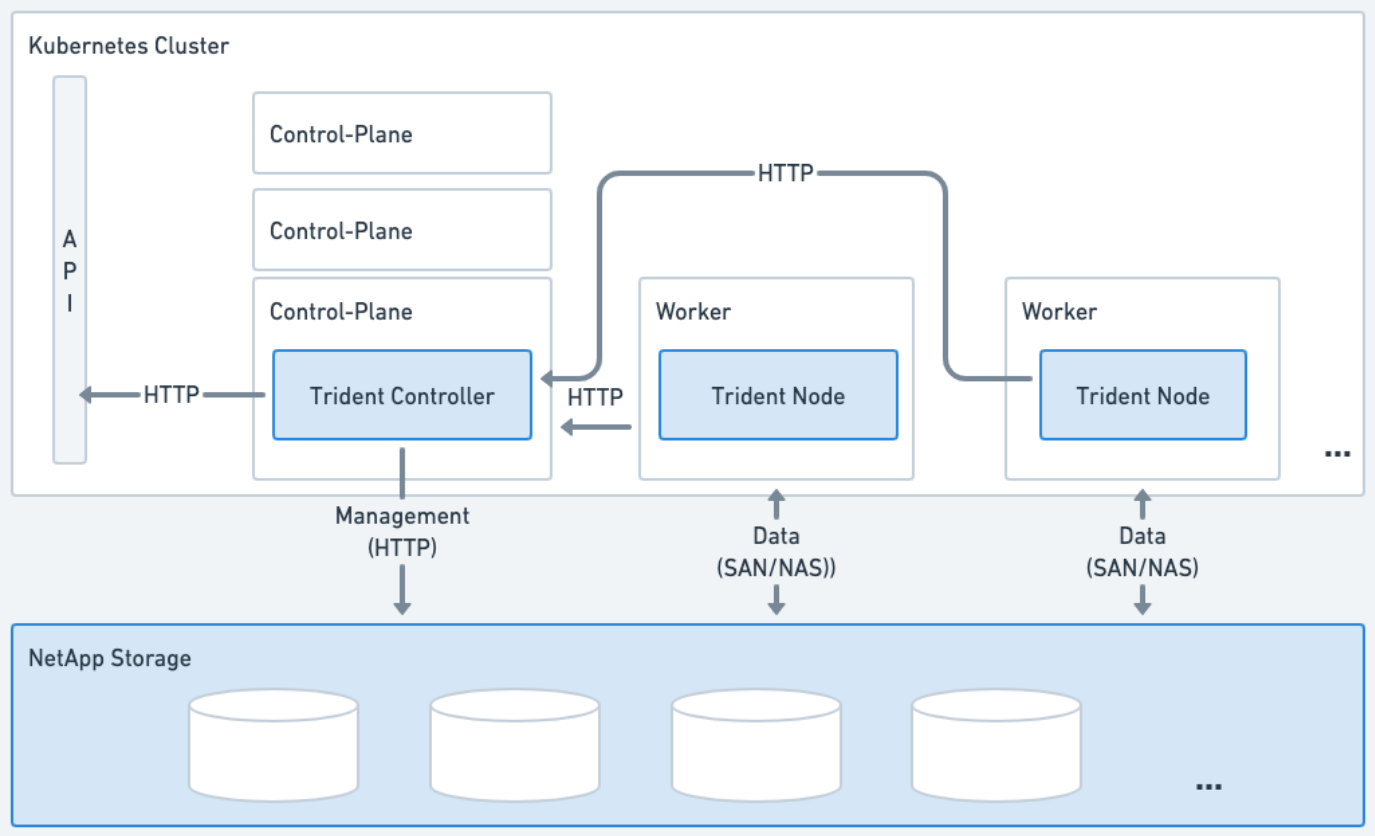


Figura 1. Astra Trident implantado no cluster Kubernetes

Pod do controlador Trident

O Pod do controlador Trident é um único Pod que executa o plug-in do controlador CSI.

- Responsável pelo provisionamento e gerenciamento de volumes no storage NetApp
- Gerenciado por uma implantação do Kubernetes
- Pode ser executado no plano de controle ou nos nós de trabalho, dependendo dos parâmetros de instalação.

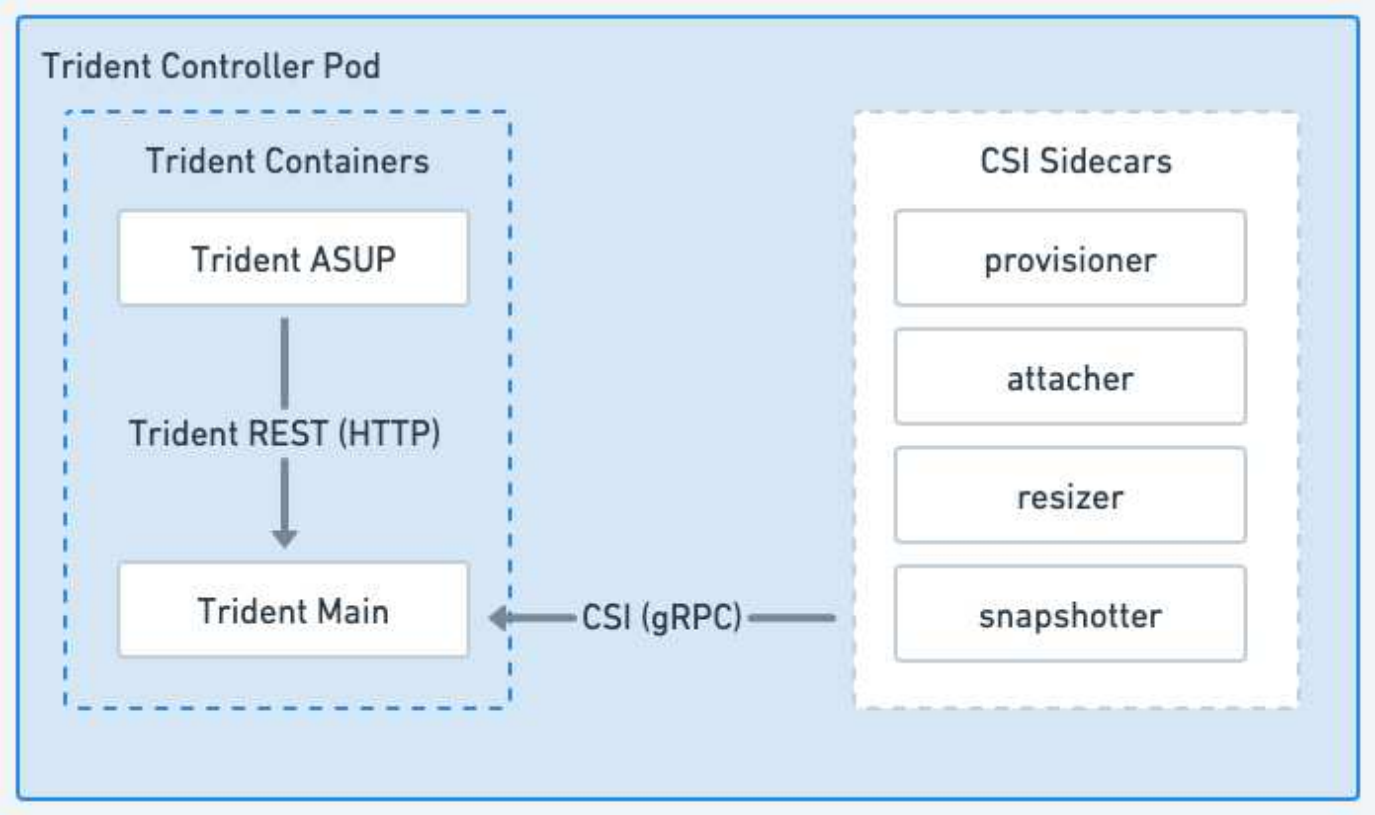


Figura 2. Diagrama do pod do controlador Trident

Pods do nó Trident

Os pods de nó Trident são pods privilegiados que executam o plug-in do nó CSI.

- Responsável pela montagem e desmontagem do armazenamento dos pods em execução no host
- Gerenciado por um DaemonSet Kubernetes
- Deve ser executado em qualquer nó que montar o storage NetApp

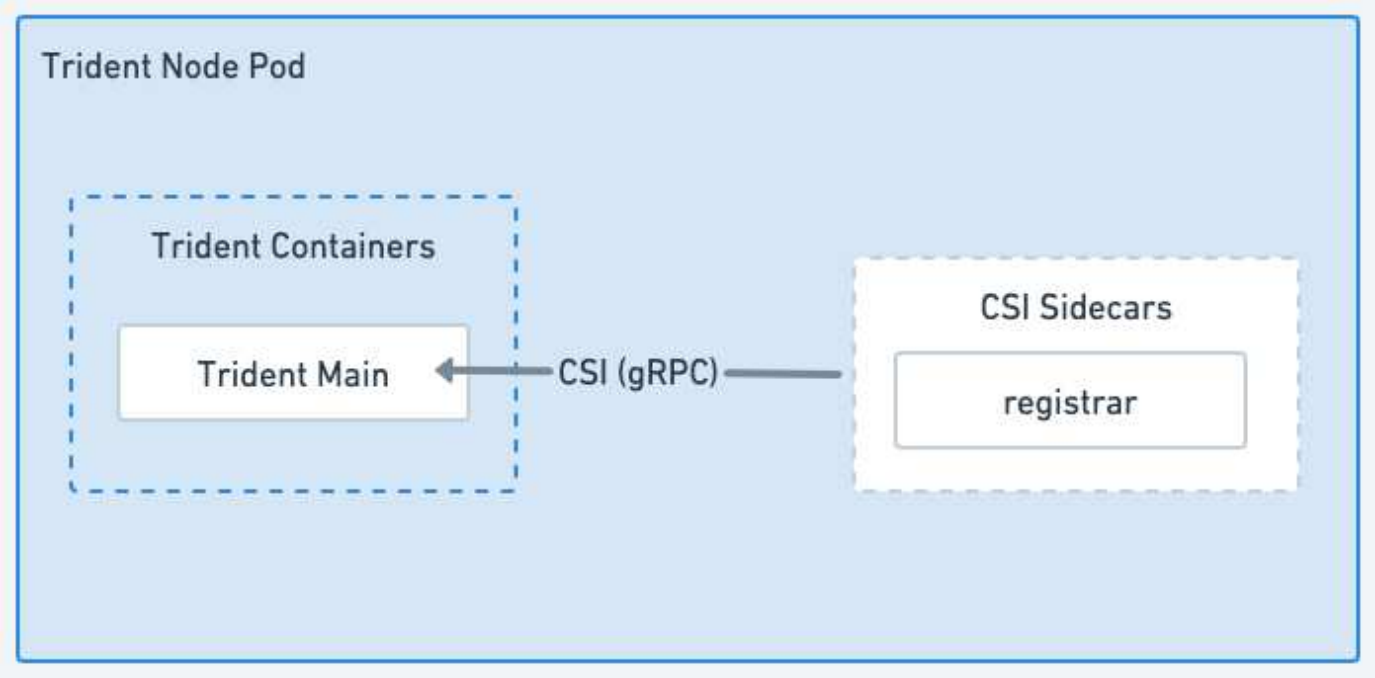


Figura 3. Diagrama do pod do nó Trident

Arquiteturas de cluster Kubernetes compatíveis

O Astra Trident é compatível com as seguintes arquiteturas de Kubernetes:

Arquiteturas de cluster do Kubernetes	Suportado	Instalação predefinida
Único mestre, computação	Sim	Sim
Mestre múltiplo, computação	Sim	Sim
Mestre `etcd`, , computação	Sim	Sim
Mestre, infraestrutura, computação	Sim	Sim

Conceitos

Provisionamento

O provisionamento no Astra Trident tem duas fases primárias. A primeira fase associa uma classe de armazenamento ao conjunto de conjuntos de armazenamento de back-end adequados e ocorre como uma preparação necessária antes do provisionamento. A segunda fase inclui a própria criação de volume e requer a escolha de um pool de armazenamento daqueles associados à classe de armazenamento do volume pendente.

Associação de classe de armazenamento

A associação de pools de storage de back-end a uma classe de armazenamento depende dos atributos solicitados da classe de armazenamento e `storagePools` das listas , `additionalStoragePools` e `excludeStoragePools` . Quando você cria uma classe de storage, o Trident compara os atributos e pools

oferecidos por cada um de seus back-ends aos solicitados pela classe de storage. Se os atributos e o nome de um pool de storage corresponderem a todos os atributos e nomes de pool solicitados, o Astra Trident adicionará esse pool de storage ao conjunto de pools de storage adequados para essa classe de storage. Além disso, o Astra Trident adiciona todos os pools de storage listados na `additionalStoragePools` lista a esse conjunto, mesmo que seus atributos não atendam a todos ou a qualquer um dos atributos solicitados da classe de storage. Você deve usar a `excludeStoragePools` lista para substituir e remover pools de armazenamento de uso para uma classe de armazenamento. O Astra Trident executa um processo semelhante sempre que você adiciona um novo back-end, verificando se os pools de storage atendem às classes de storage existentes e removendo quaisquer que tenham sido marcados como excluídos.

Criação de volume

Em seguida, o Astra Trident usa as associações entre classes de storage e pools de storage para determinar onde provisionar volumes. Quando você cria um volume, o Astra Trident primeiro obtém o conjunto de pools de storage para a classe de storage desse volume e, se você especificar um protocolo para o volume, o Astra Trident removerá esses pools de storage que não podem fornecer o protocolo solicitado (por exemplo, um back-end NetApp HCI/SolidFire não poderá fornecer um volume baseado em arquivo enquanto um back-end do ONTAP nas não puder fornecer um volume baseado em bloco). O Astra Trident aleatoriza a ordem desse conjunto resultante, para facilitar uma distribuição uniforme de volumes e, em seguida, iterá-lo, tentando provisionar o volume em cada pool de storage por sua vez. Se for bem-sucedido em um, ele retorna com sucesso, registrando quaisquer falhas encontradas no processo. O Astra Trident retorna uma falha **somente se** falhar ao provisionamento em **todos** dos pools de storage disponíveis para a classe de storage e o protocolo solicitados.

Instantâneos de volume

Saiba mais sobre como o Astra Trident lida com a criação de snapshots de volume para seus drivers.

Saiba mais sobre a criação de instantâneos de volume

- Para os `ontap-nas` drivers , `ontap-san`, , `gcp-cvs` e , `azure-netapp-files` cada volume persistente (PV) é mapeado para um FlexVol. Como resultado, os snapshots de volume são criados como snapshots do NetApp. A tecnologia Snapshot da NetApp oferece mais estabilidade, escalabilidade, capacidade de recuperação e desempenho do que as tecnologias de snapshot da concorrência. Essas cópias snapshot são extremamente eficientes no tempo necessário para criá-las e no espaço de storage.
- Para `ontap-nas-flexgroup` o condutor, cada volume persistente (PV) é mapeado para um FlexGroup. Como resultado, os snapshots de volume são criados como snapshots do NetApp FlexGroup. A tecnologia Snapshot da NetApp oferece mais estabilidade, escalabilidade, capacidade de recuperação e desempenho do que as tecnologias de snapshot da concorrência. Essas cópias snapshot são extremamente eficientes no tempo necessário para criá-las e no espaço de storage.
- Para `ontap-san-economy` o driver, os PVS mapeiam para LUNs criados em FlexVols compartilhados. VolumeSnapshots de PVS são obtidos executando FlexClones do LUN associado. Com a tecnologia ONTAP FlexClone, é possível criar cópias dos maiores conjuntos de dados quase instantaneamente. As cópias compartilham blocos de dados com os pais, não consumindo storage, exceto o necessário para os metadados.
- Para `solidfire-san` o driver, cada PV mapeia para um LUN criado no cluster do software/NetApp HCI do NetApp Element. VolumeSnapshots são representados por instantâneos de elementos do LUN subjacente. Esses snapshots são cópias pontuais e ocupam apenas um pequeno espaço e recursos do sistema.
- Ao trabalhar com `ontap-nas` os drivers e `ontap-san`, os snapshots do ONTAP são cópias pontuais do FlexVol e consomem espaço no próprio FlexVol. Isso pode resultar na quantidade de espaço gravável no

volume para reduzir com o tempo, à medida que os snapshots são criados/programados. Uma maneira simples de lidar com isso é aumentar o volume redimensionando pelo Kubernetes. Outra opção é excluir snapshots que não são mais necessários. Quando um VolumeSnapshot criado por meio do Kubernetes é excluído, o Astra Trident excluirá o snapshot associado do ONTAP. Os snapshots do ONTAP que não foram criados pelo Kubernetes também podem ser excluídos.

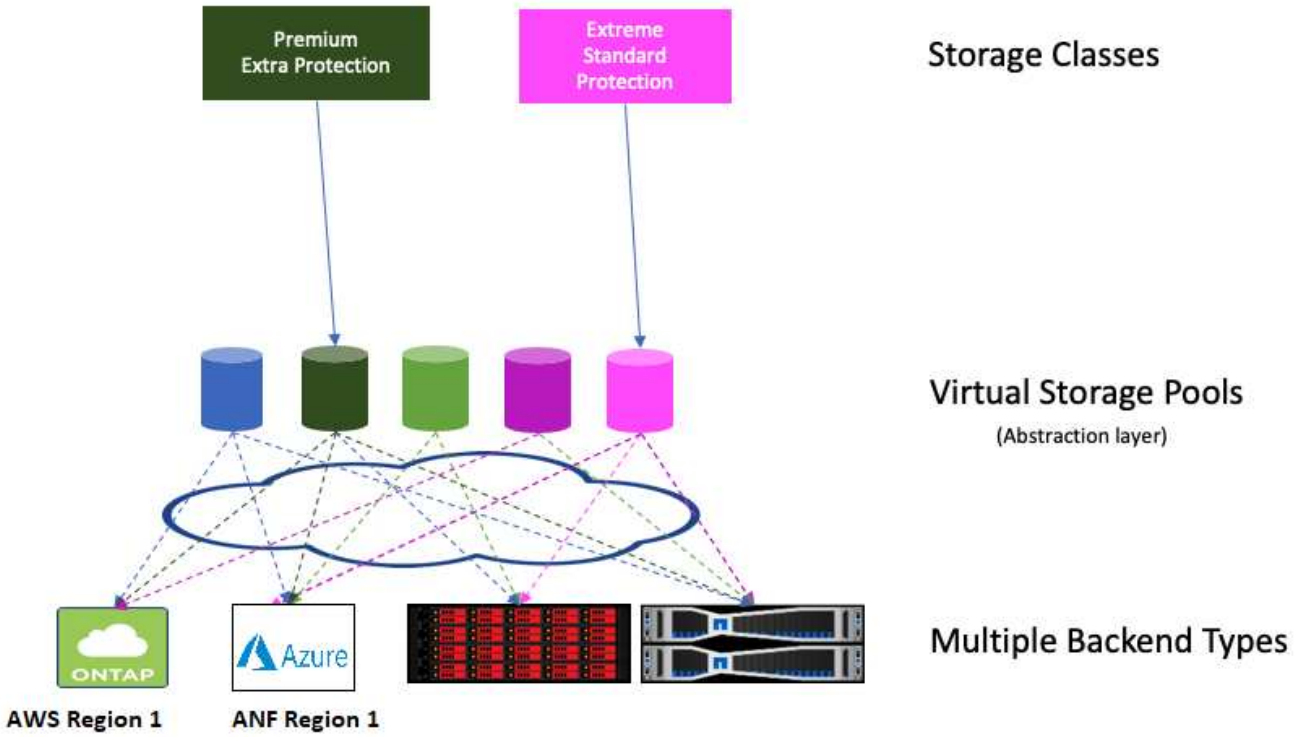
Com o Astra Trident, você pode usar o VolumeSnapshots para criar novos PVS a partir deles. A criação de PVS a partir desses snapshots é realizada usando a tecnologia FlexClone para backends ONTAP e CVS compatíveis. Ao criar um PV a partir de um instantâneo, o volume de apoio é um FlexClone do volume pai do instantâneo. O `solidfire-san` driver usa clones de volume do software Element para criar PVS a partir de snapshots. Aqui ele cria um clone a partir do snapshot do elemento.

Pools virtuais

Os pools virtuais fornecem uma camada de abstração entre os back-ends de storage do Astra Trident e o Kubernetes `StorageClasses`. Eles permitem que um administrador defina aspetos, como localização, desempenho e proteção para cada back-end de uma maneira comum e independente de back-end, sem `StorageClass` especificar qual backend físico, pool de back-end ou tipo de back-end usar para atender aos critérios desejados.

Saiba mais sobre pools virtuais

O administrador de storage pode definir pools virtuais em qualquer um dos backends do Astra Trident em um arquivo de definição JSON ou YAML.



Qualquer aspeto especificado fora da lista de pools virtuais é global para o back-end e se aplicará a todos os

pools virtuais, enquanto cada pool virtual pode especificar um ou mais aspetos individualmente (substituindo quaisquer aspetos globais de back-end).

- Ao definir pools virtuais, não tente reorganizar a ordem dos pools virtuais existentes em uma definição de back-end.
- Aconselhamos a não modificar atributos para um pool virtual existente. Você deve definir um novo pool virtual para fazer alterações.

A maioria dos aspetos são especificados em termos específicos de back-end. Fundamentalmente, os valores de aspeto não são expostos fora do driver do back-end e não estão disponíveis para correspondência em `StorageClasses`. em vez disso, o administrador define um ou mais rótulos para cada pool virtual. Cada rótulo é um par chave:valor, e os rótulos podem ser comuns em backends exclusivos. Assim como aspetos, os rótulos podem ser especificados por pool ou globais para o back-end. Ao contrário de aspetos, que têm nomes e valores predefinidos, o administrador tem total discrição para definir chaves de rótulo e valores conforme necessário. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

A `StorageClass` identifica qual pool virtual usar fazendo referência aos rótulos dentro de um parâmetro seletor. Os seletores de pool virtual suportam os seguintes operadores:

Operador	Exemplo	O valor do rótulo de um pool deve:
`=`	desempenho superior	Correspondência
`!=`	performance! extrema	Não corresponde
`in`	localização em (leste, oeste)	Esteja no conjunto de valores
`notin`	notificação de desempenho (prata, bronze)	Não estar no conjunto de valores
`<key>`	proteção	Existe com qualquer valor
`!<key>`	!proteção	Não existe

Grupos de acesso de volume

Saiba mais sobre como o Astra Trident usa "grupos de acesso de volume".

Ignore esta seção se você estiver usando CHAP, que é recomendado para simplificar o gerenciamento e evitar o limite de escala descrito abaixo. Além disso, se você estiver usando Astra Trident no modo CSI, você pode ignorar esta seção. O Astra Trident usa o CHAP quando instalado como um provisionador aprimorado de CSI.

Saiba mais sobre grupos de acesso de volume

O Astra Trident pode usar grupos de acesso a volumes para controlar o acesso aos volumes provisionados. Se o CHAP estiver desativado, ele espera encontrar um grupo de acesso chamado `trident`, a menos que você especifique um ou mais IDs de grupo de acesso na configuração.

Embora o Astra Trident associe novos volumes aos grupos de acesso configurados, ele não cria nem gerencia grupos de acesso por conta própria. Os grupos de acesso devem existir antes que o back-end de storage seja adicionado ao Astra Trident e precisam conter as IQNs iSCSI de todos os nós do cluster do Kubernetes que poderiam potencialmente montar os volumes provisionados por esse back-end. Na maioria das instalações, isso inclui cada nó de trabalho no cluster.

Para clusters de Kubernetes com mais de 64 nós, você deve usar vários grupos de acesso. Cada grupo de acesso pode conter até 64 IQNs e cada volume pode pertencer a quatro grupos de acesso. Com o máximo de quatro grupos de acesso configurados, qualquer nó em um cluster de até 256 nós de tamanho poderá acessar qualquer volume. Para obter os limites mais recentes dos grupos de acesso de volume, "aqui" consulte a .

Se você estiver modificando a configuração de uma que esteja usando o grupo de acesso padrão `trident` para outra que também use outras, inclua a ID do `trident` grupo de acesso na lista.

Início rápido para Astra Trident

Você pode instalar o Astra Trident e começar a gerenciar recursos de storage em apenas algumas etapas. Antes de começar, reveja "Requisitos do Astra Trident"o .

Para Docker, "Astra Trident para Docker"consulte .

1 Instale o Astra Trident

O Astra Trident oferece vários métodos e modos de instalação otimizados para uma variedade de ambientes e organizações.

"Instale o Astra Trident"

2 Prepare o nó de trabalho

Todos os nós de trabalho no cluster do Kubernetes precisam ser capazes de montar os volumes provisionados para os pods.

"Prepare o nó de trabalho"

3 Crie um backend

Um back-end define a relação entre o Astra Trident e um sistema de storage. Ele diz ao Astra Trident como se comunicar com esse sistema de storage e como o Astra Trident deve provisionar volumes a partir dele.

"Configurar um back-end" para o seu sistema de storage

4 Crie um Kubernetes StorageClass

O objeto Kubernetes StorageClass especifica o Astra Trident como o provisionador e permite que você crie uma classe de storage para provisionar volumes com atributos personalizáveis. O Astra Trident cria uma classe de storage correspondente para objetos Kubernetes que especificam o provisionador do Astra Trident.

"Crie uma classe de armazenamento"

5 Provisionar um volume

Um *Persistentvolume* (PV) é um recurso de armazenamento físico provisionado pelo administrador de cluster em um cluster do Kubernetes. O *PersistentVolumeClaim* (PVC) é um pedido de acesso ao Persistentvolume

no cluster.

Crie um Persistentvolume (PV) e um PersistentVolumeClaim (PVC) que use o Kubernetes StorageClass configurado para solicitar acesso ao PV. Em seguida, pode montar o PV num pod.

"Provisionar um volume"

O que se segue?

Agora você pode adicionar backends adicionais, gerenciar classes de armazenamento, gerenciar backends e executar operações de volume.

Requisitos

Antes de instalar o Astra Trident, você deve analisar esses requisitos gerais de sistema. Backends específicos podem ter requisitos adicionais.

Informações essenciais sobre o Astra Trident

Você deve ler as seguintes informações críticas sobre o Astra Trident.

** informações essenciais sobre o Astra Trident **

- O Kubernetes 1,29 agora é compatível com o Trident. Atualize o Astra Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. O Astra Trident recomenda o uso `find_multipaths: no` desde o lançamento de 21,07.

Frontens suportados (orquestradores)

O Astra Trident é compatível com vários mecanismos de contêiner e orquestradores, incluindo os seguintes:

- Anthos On-Prem (VMware) e Anthos em bare metal 1,16
- Kubernetes 1,23 - 1,29
- OpenShift 4,10 - 4,15

O operador Trident é suportado com estas versões:

- Anthos On-Prem (VMware) e Anthos em bare metal 1,16
- Kubernetes 1,23 - 1,29
- OpenShift 4,10 - 4,15

O Astra Trident também trabalha com uma série de outras ofertas do Kubernetes totalmente gerenciadas e autogeridas, incluindo o Google Kubernetes Engine (GKE), o Amazon Elastic Kubernetes Services (EKS), o

O Astra Trident e o ONTAP podem ser usados como fornecedores de storage do "KubeVirt".

> Antes de atualizar um cluster do Kubernetes do 1,24 para o 1,25 ou posterior que tenha o Astra Trident instalado, "Atualize uma instalação do Helm"consulte a .

Backends suportados (armazenamento)

Para usar o Astra Trident, você precisa de um ou mais dos seguintes back-ends compatíveis:

- Amazon FSX para NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Cloud Volumes Service para GCP
- FAS/AFF/Selecione 9,5 ou posterior
- NetApp All SAN Array (ASA)
- Software NetApp HCI/Element 11 ou superior

Requisitos de recursos

A tabela abaixo resume os recursos disponíveis com esta versão do Astra Trident e as versões do Kubernetes compatíveis.

Recurso	Versão do Kubernetes	É necessário ter portões?
Astra Trident	1,23 - 1,29	Não
Instantâneos de volume	1,23 - 1,29	Não
PVC a partir de instantâneos de volume	1,23 - 1,29	Não
Redimensionamento iSCSI PV	1,23 - 1,29	Não
ONTAP bidirectional CHAP	1,23 - 1,29	Não
Políticas de exportação dinâmica	1,23 - 1,29	Não
Operador Trident	1,23 - 1,29	Não
Topologia de CSI	1,23 - 1,29	Não

Sistemas operacionais de host testados

Embora o Astra Trident não seja oficialmente compatível com sistemas operacionais específicos, sabe-se que os seguintes itens funcionam:

- Versões do RedHat CoreOS (RHCOS) suportadas pela OpenShift Container Platform (AMD64 e ARM64)

- RHEL 8 OU SUPERIOR (AMD64 E ARM64)

  O NVMe/TCP requer o RHEL 9 ou posterior.

- Ubuntu 22,04 ou posterior (AMD64 e ARM64)
- Windows Server 2019 (AMD64 bits)

Por padrão, o Astra Trident é executado em um contentor e, portanto, será executado em qualquer trabalhador Linux. No entanto, esses funcionários precisam ser capazes de montar os volumes que o Astra Trident fornece usando o cliente NFS padrão ou iniciador iSCSI, dependendo dos backends que você está usando.

O `tridentctl` utilitário também é executado em qualquer uma dessas distribuições do Linux.

Configuração de host

Todos os nós de trabalho no cluster do Kubernetes precisam ser capazes de montar os volumes provisionados para os pods. Para preparar os nós de trabalho, é necessário instalar ferramentas NFS, iSCSI ou NVMe com base na seleção de driver.

"Prepare o nó de trabalho"

Configuração do sistema de storage

O Astra Trident pode exigir alterações em um sistema de storage antes que uma configuração de back-end o use.

"Configurar backends"

Portas Astra Trident

O Astra Trident requer acesso a portas específicas para comunicação.

"Portas Astra Trident"

Imagens de contêineres e versões correspondentes do Kubernetes

Para instalações com conexão de ar, a lista a seguir é uma referência das imagens de contêiner necessárias para instalar o Astra Trident. Use o `tridentctl images` comando para verificar a lista de imagens de contentor necessárias.

Versão do Kubernetes	Imagem do recipiente
v1.23.0	• docker.io/NetApp/Trident:24.02.0 • docker.io/NetApp/Trident-AutoSupport:24,02 • provisionador do registry.k8s.io/sig-storage/csi:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1.9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.10.0 • docker.io/NetApp/Trident-operador:24.02.0 (opcional)
v1.24.0	• docker.io/NetApp/Trident:24.02.0 • docker.io/NetApp/Trident-AutoSupport:24,02 • provisionador do registry.k8s.io/sig-storage/csi:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1.9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.10.0 • docker.io/NetApp/Trident-operador:24.02.0 (opcional)
v1.25.0	• docker.io/NetApp/Trident:24.02.0 • docker.io/NetApp/Trident-AutoSupport:24,02 • provisionador do registry.k8s.io/sig-storage/csi:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1.9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.10.0 • docker.io/NetApp/Trident-operador:24.02.0 (opcional)

Versão do Kubernetes	Imagem do recipiente
v1.26.0	• docker.io/NetApp/Trident:24.02.0 • docker.io/NetApp/Trident-AutoSupport:24,02 • provisionador do registry.k8s.io/sig-storage/csi:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1.9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.10.0 • docker.io/NetApp/Trident-operador:24.02.0 (opcional)
v1.27.0	• docker.io/NetApp/Trident:24.02.0 • docker.io/NetApp/Trident-AutoSupport:24,02 • provisionador do registry.k8s.io/sig-storage/csi:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1.9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.10.0 • docker.io/NetApp/Trident-operador:24.02.0 (opcional)
v1.28.0	• docker.io/NetApp/Trident:24.02.0 • docker.io/NetApp/Trident-AutoSupport:24,02 • provisionador do registry.k8s.io/sig-storage/csi:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1.9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.10.0 • docker.io/NetApp/Trident-operador:24.02.0 (opcional)

Versão do Kubernetes	Imagem do recipiente
v1.29.0	• docker.io/NetApp/Trident:24.02.0 • docker.io/NetApp/Trident-AutoSupport:24,02 • provisionador do registry.k8s.io/sig-storage/csi:v4,0.0 • registry.k8s.io/sig-storage/csi-attacher:v4,5.0 • registry.k8s.io/sig-storage/csi-resizer:v1.9.3 • registry.k8s.io/sig-storage/csi-snapshotter:v6,3.3 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2.10.0 • docker.io/NetApp/Trident-operador:24.02.0 (opcional)

Instale o Astra Trident

Saiba mais sobre a instalação do Astra Trident

Para garantir que o Astra Trident possa ser instalado em uma ampla variedade de ambientes e organizações, o NetApp oferece várias opções de instalação. Você pode instalar o Astra Trident usando o operador Trident (manualmente ou usando o Helm) ou com `tridentctl`o . Este tópico fornece informações importantes para selecionar o processo de instalação certo para você.

Informações críticas sobre o Astra Trident 24,02

Você deve ler as seguintes informações críticas sobre o Astra Trident.

** informações essenciais sobre o Astra Trident **

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Antes de começar

Independentemente do seu caminho de instalação, você deve ter:

- Privileges completo para um cluster Kubernetes compatível com execução de uma versão compatível do Kubernetes e requisitos de recursos habilitados. Reveja a "requisitos" para obter detalhes.
- Acesso a um sistema de storage NetApp compatível.
- Funcionalidade de montar volumes de todos os nós de trabalho do Kubernetes.
- Um host Linux com `kubectl` (ou `oc`, se você estiver usando o OpenShift) instalado e configurado para gerenciar o cluster do Kubernetes que deseja usar.
- A `KUBECONFIG` variável de ambiente configurada para apontar para a configuração do cluster do Kubernetes.
- Se você estiver usando o Kubernetes com Docker Enterprise "Siga os passos para ativar o acesso CLI", .

Se você não se familiarizou com o "conceitos básicos", agora é um grande momento para fazer isso.

Escolha o método de instalação

Selecione o método de instalação correto para você. Você também deve rever as considerações "movendo-se entre métodos"antes de tomar sua decisão.

Utilizando o operador Trident

Seja implantando manualmente ou usando o Helm, o operador Trident é uma ótima maneira de simplificar a instalação e gerenciar dinamicamente os recursos do Astra Trident. Você pode até mesmo "Personalize a implantação do seu operador Trident" usar os atributos no `TridentOrchestrator` recurso personalizado (CR).

Os benefícios de usar o operador Trident incluem:

Astra Trident Object creation

O operador Trident cria automaticamente os seguintes objetos para a versão do Kubernetes.

- ServiceAccount para o operador
- ClusterRole e ClusterRoleBinding para o ServiceAccount
- PodSecurityPolicy dedicada (para Kubernetes 1,25 e versões anteriores)
- O próprio operador

Accountability de recursos

O operador Trident com escopo de cluster gerencia recursos associados a uma instalação do Astra Trident no nível do cluster. Isso atenua erros que podem ser causados ao manter recursos com escopo de cluster usando um operador com escopo de namespace. Isso é essencial para a auto-cura e correção.

** capacidade de autorrecuperação **

O operador monitora a instalação do Astra Trident e toma ativamente medidas para resolver problemas, como quando a implantação é excluída ou se for modificada acidentalmente. É criado um `trident-operator-<generated-id>` pod que associa `TridentOrchestrator` um CR a uma instalação do Astra Trident. Isso garante que haja apenas uma instância do Astra Trident no cluster e controla sua configuração, garantindo que a instalação seja idempotente. Quando as alterações são feitas na instalação (como, por exemplo, a exclusão do daemonset de implantação ou nó), o operador as identifica e as corrige individualmente.

** atualizações fáceis para existente**

Você pode facilmente atualizar uma implantação existente com o operador. Você só precisa editar o `TridentOrchestrator` CR para fazer atualizações em uma instalação.

Por exemplo, considere um cenário em que você precisa habilitar o Astra Trident para gerar logs de depuração. Para fazer isso, corrija o `TridentOrchestrator` para definir `spec.debug` como `true`:

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge
-p '{"spec":{"debug":true}}'
```

Após `TridentOrchestrator` a atualização, o operador processa as atualizações e corrige a instalação existente. Isso pode acionar a criação de novos pods para modificar a instalação de acordo.

** Clean restablation**

O operador Trident com escopo de cluster permite a remoção limpa de recursos com escopo de cluster. Os usuários podem desinstalar completamente o Astra Trident e reinstalar facilmente.

** handling de atualização automática do Kubernetes**

Quando a versão do Kubernetes do cluster é atualizada para uma versão compatível, a operadora atualiza uma instalação existente do Astra Trident automaticamente e a altera para garantir que ela atenda aos requisitos da versão do Kubernetes.

> ⓘ Se o cluster for atualizado para uma versão não suportada, o operador impede a instalação do Astra Trident. Se o Astra Trident já tiver sido instalado com a operadora, um aviso será exibido para indicar que o Astra Trident está instalado em uma versão Kubernetes não suportada.

** gerenciamento de clusters do Kubernetes usando o BlueXP (anteriormente conhecido como Cloud Manager) **

Com "Astra Trident usando BlueXP "o , você pode atualizar para a versão mais recente do Astra Trident, adicionar e gerenciar classes de storage, conectá-las a ambientes de trabalho e fazer backup de volumes persistentes usando o Cloud Backup Service. O BlueXP oferece suporte à implantação do Astra Trident usando o operador Trident, manualmente ou usando o Helm.

Utilização `tridentctl`

Se você tiver uma implantação existente que deve ser atualizada ou se você estiver procurando personalizar altamente sua implantação, considere o . Esse é o método convencional de implantação do Astra Trident.

Você pode gerar os manifestos para recursos do Trident. Isso inclui a implantação, o daemonset, a conta de serviço e a função de cluster que o Astra Trident cria como parte de sua instalação.

A partir da versão 22,04, as chaves AES não serão mais regeneradas sempre que o Astra Trident for instalado. Com este lançamento, o Astra Trident instalará um novo objeto secreto que persiste em todas as instalações. Isso significa que `tridentctl` no 22,04 pode desinstalar versões anteriores do Trident, mas versões anteriores não podem desinstalar instalações do 22,04. Selecione a instalação apropriada *method*.

Escolha o modo de instalação

Determine seu processo de implantação com base no *modo de instalação* (padrão, Offline ou remoto) exigido pela sua organização.

Instalação padrão

Essa é a maneira mais fácil de instalar o Astra Trident e funciona para a maioria dos ambientes que não impõem restrições de rede. O modo de instalação padrão usa Registros padrão para armazenar (`registry.k8s.io`imagens Trident (`docker.io`) e CSI necessárias.

Quando você usa o modo padrão, o instalador do Astra Trident:

- Obtém as imagens de contentor pela Internet
- Cria um daemonset de implantação ou nó, que ativa pods do Astra Trident em todos os nós qualificados no cluster do Kubernetes

Instalação offline

O modo de instalação off-line pode ser necessário em um local seguro ou protegido. Nesse cenário, você pode criar um único Registro privado espelhado ou dois Registros espelhados para armazenar imagens Trident e CSI necessárias.

Independentemente da configuração do Registro, as imagens CSI devem residir em um Registro.

Instalação remota

Aqui está uma visão geral de alto nível do processo de instalação remota:

- Implante a versão apropriada do `kubectl` na máquina remota de onde você deseja implantar o Astra Trident.
- Copie os arquivos de configuração do cluster do Kubernetes e defina a `KUBECONFIG` variável de ambiente na máquina remota.
- Inicie um `kubectl get nodes` comando para verificar se você pode se conetar ao cluster do Kubernetes necessário.
- Conclua a implementação a partir da máquina remota utilizando as etapas de instalação padrão.

Selecione o processo com base no seu método e modo

Depois de tomar suas decisões, selecione o processo apropriado.

Método	Modo de instalação
Operador Trident (manualmente)	"Instalação padrão" "Instalação offline"
Operador Trident (Helm)	"Instalação padrão" "Instalação offline"
`tridentctl`	"Instalação padrão ou offline"

Movendo-se entre os métodos de instalação

Você pode decidir alterar seu método de instalação. Antes de fazer isso, considere o seguinte:

- Sempre use o mesmo método para instalar e desinstalar o Astra Trident. Se você implantou com `tridentctl`o , use a versão apropriada `tridentctl` do binário para desinstalar o Astra Trident. Da mesma forma, se você estiver implantando com o operador, edite o `TridentOrchestrator` CR e configure `spec.uninstall=true` para desinstalar o Astra Trident.
- Se você tiver uma implantação baseada em operador que deseja remover e usar `tridentctl` para implantar o Astra Trident, primeiro edite `TridentOrchestrator` e configure `spec.uninstall=true` para desinstalar o Astra Trident. Em seguida, exclua `TridentOrchestrator` e a implantação do operador. Você pode instalar usando `tridentctl`o .
- Se você tiver uma implantação manual baseada no operador e quiser usar a implantação do operador Trident baseada no Helm, desinstale manualmente o operador primeiro e execute a instalação do Helm. Isso permite que o Helm implante o operador Trident com as etiquetas e anotações necessárias. Se você não fizer isso, sua implantação de operador Trident baseada em Helm falhará com erro de validação de rótulo e erro de validação de anotação. Se você tem uma `tridentctl` implantação baseada em -, você pode usar a implantação baseada em Helm sem problemas.

Outras opções de configuração conhecidas

Ao instalar o Astra Trident em produtos do portfólio VMware Tanzu:

- O cluster precisa dar suporte a workloads privilegiados.
- A `--kubelet-dir` bandeira deve ser definida para a localização do diretório kubelet. Por padrão, isso é `/var/vcap/data/kubelet`.

  Especificar a localização do kubelet usando `--kubelet-dir` é conhecido por funcionar para o Operador Trident, Helm e `tridentctl` implantações.

Instale usando o operador Trident

Implantar manualmente o operador Trident (modo padrão)

Você pode implantar manualmente o operador Trident para instalar o Astra Trident. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra

Trident não são armazenadas em um Registro privado. Se tiver um registo de imagens privado, utilize o "processo para implantação off-line".

Informações críticas sobre o Astra Trident 24,02

Você deve ler as seguintes informações críticas sobre o Astra Trident.

** informações essenciais sobre o Astra Trident **

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante manualmente o operador Trident e instale o Trident

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Antes de iniciar a instalação, faça login no host Linux e verifique se ele está gerenciando um trabalho e "Cluster compatível com Kubernetes"se você tem o Privileges necessário.

> Com OpenShift, use `oc` em vez de `kubectl` em todos os exemplos que se seguem, e faça login como **system:admin** primeiro executando `oc login -u system:admin` ou `oc login -u kube-admin`.

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique o Privileges do administrador do cluster:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verifique se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Passo 1: Baixe o pacote de instalação do Trident

O pacote de instalação do Astra Trident contém tudo o que você precisa para implantar o operador Trident e instalar o Astra Trident. Baixe e extraia a versão mais recente do instalador do Trident a partir "A seção *assets* no GitHub"do .

```
wget https://github.com/NetApp/trident/releases/download/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.tar.gz
cd trident-installer
```

Passo 2: Crie o `TridentOrchestrator` CRD

Crie a `TridentOrchestrator` Definição de recurso personalizada (CRD). Você criará um `TridentOrchestrator` recurso personalizado mais tarde. Use a versão apropriada do CRD YAML em `deploy/crds` para criar o `TridentOrchestrator` CRD.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

Etapa 3: Implante o operador Trident

O instalador do Astra Trident fornece um arquivo de pacote que pode ser usado para instalar o operador e criar objetos associados. O arquivo de pacote é uma maneira fácil de implantar o operador e instalar o Astra Trident usando uma configuração padrão.

- Para clusters que executam o Kubernetes 1,24 ou anterior, `bundle_pre_1_25.yaml` use o .

- Para clusters que executam o Kubernetes 1,25 ou posterior, `bundle_post_1_25.yaml` use o .

Antes de começar

- Por padrão, o instalador do Trident implanta o operador no `trident` namespace. Se o `trident` namespace não existir, crie-o usando:

```
kubectl apply -f deploy/namespace.yaml
```

- Para implantar o operador em um namespace diferente do `trident` namespace, atualizar `serviceaccount.yaml` `clusterrolebinding.yaml` `operator.yaml` e gerar o arquivo do pacote usando o `kustomization.yaml`.

  a. Crie o `kustomization.yaml` usando o seguinte comando onde *<bundle.yaml>* está `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` baseado na sua versão do Kubernetes.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. Compile o pacote usando o seguinte comando onde *<bundle.yaml>* está `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` baseado na sua versão do Kubernetes.

  ```
  kubectl kustomize deploy/ > deploy/<bundle.yaml>
  ```

Passos

1. Crie os recursos e implante o operador:

```
kubectl create -f deploy/<bundle.yaml>
```

2. Verifique se o operador, a implantação e as replicasets foram criados.

```
kubectl get all -n <operator-namespace>
```

> Deve haver apenas **uma instância** do operador em um cluster do Kubernetes. Não crie várias implantações do operador Trident.

Passo 4: Crie o `TridentOrchestrator` e instale o Trident

Agora você pode criar e instalar o `TridentOrchestrator` Astra Trident. Opcionalmente, você pode "Personalize a instalação do Trident" usar os atributos na `TridentOrchestrator` especificação.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:24.02
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:24.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v24.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verifique a instalação

Existem várias maneiras de verificar sua instalação.

`TridentOrchestrator`Usando o status

O status de `TridentOrchestrator` indica se a instalação foi bem-sucedida e exibe a versão do Trident

instalado. Durante a instalação, o status das `TridentOrchestrator` alterações de `Installing` para `Installed`. Se você observar o `Failed` status e o operador não conseguir recuperar sozinho, "verifique os logs".

Estado	Descrição
A instalar	O operador está instalando o Astra Trident usando este `TridentOrchestrator` CR.
Instalado	O Astra Trident foi instalado com sucesso.
Desinstalação	O operador está desinstalando o Astra Trident, `spec.uninstall=true` porque .
Desinstalado	O Astra Trident foi desinstalado.
Falha	O operador não pôde instalar, corrigir, atualizar ou desinstalar o Astra Trident; o operador tentará recuperar automaticamente deste estado. Se este estado persistir, será necessário resolver o problema.
A atualizar	O operador está atualizando uma instalação existente.
Erro	O `TridentOrchestrator` não é utilizado. Outro já existe.

Usando o status de criação do pod

Você pode confirmar se a instalação do Astra Trident foi concluída analisando o status dos pods criados:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

Utilização `tridentctl`

Você pode usar `tridentctl` para verificar a versão do Astra Trident instalada.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+
```

Implantar manualmente o operador Trident (modo off-line)

Você pode implantar manualmente o operador Trident para instalar o Astra Trident. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra Trident são armazenadas em um Registro privado. Se não tiver um registo de imagens privado, utilize o "processo para implantação padrão".

Informações críticas sobre o Astra Trident 24,02

Você deve ler as seguintes informações críticas sobre o Astra Trident.

informações essenciais sobre o Astra Trident

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante manualmente o operador Trident e instale o Trident

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Faça login no host Linux e verifique se ele está gerenciando um trabalho e "Cluster compatível com Kubernetes"que você tem o Privileges necessário.

Com OpenShift, use `oc` em vez de `kubectl` em todos os exemplos que se seguem, e faça login como **system:admin** primeiro executando `oc login -u system:admin` ou `oc login -u kube-admin`.

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique o Privileges do administrador do cluster:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verifique se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Passo 1: Baixe o pacote de instalação do Trident

O pacote de instalação do Astra Trident contém tudo o que você precisa para implantar o operador Trident e instalar o Astra Trident. Baixe e extraia a versão mais recente do instalador do Trident a partir "A seção *assets* no GitHub"do .

```
wget https://github.com/NetApp/trident/releases/download/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.tar.gz
cd trident-installer
```

Passo 2: Crie o `TridentOrchestrator` CRD

Crie a `TridentOrchestrator` Definição de recurso personalizada (CRD). Você criará um `TridentOrchestrator` recurso personalizado mais tarde. Use a versão apropriada do CRD YAML em `deploy/crds` para criar o `TridentOrchestrator` CRD:

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

Etapa 3: Atualize a localização do Registro no operador

No `/deploy/operator.yaml`, atualize `image: docker.io/netapp/trident-operator:24.02.0` para refletir a localização do registo de imagens. O "Imagens de Trident e CSI" pode estar localizado em um Registro ou Registros diferentes, mas todas as imagens CSI devem estar localizadas no mesmo Registro. Por exemplo:

- `image: <your-registry>/trident-operator:24.02.0` se todas as suas imagens estiverem localizadas em um registro.

- `image: <your-registry>/netapp/trident-operator:24.02.0` Se a sua imagem Trident estiver localizada num registo diferente das suas imagens CSI.

Etapa 4: Implante o operador Trident

O instalador do Astra Trident fornece um arquivo de pacote que pode ser usado para instalar o operador e criar objetos associados. O arquivo de pacote é uma maneira fácil de implantar o operador e instalar o Astra Trident usando uma configuração padrão.

- Para clusters que executam o Kubernetes 1,24 ou anterior, `bundle_pre_1_25.yaml` use o .
- Para clusters que executam o Kubernetes 1,25 ou posterior, `bundle_post_1_25.yaml` use o .

Antes de começar

- Por padrão, o instalador do Trident implanta o operador no `trident` namespace. Se o `trident` namespace não existir, crie-o usando:

```
kubectl apply -f deploy/namespace.yaml
```

- Para implantar o operador em um namespace diferente do `trident` namespace, atualizar `serviceaccount.yaml` `clusterrolebinding.yaml` `operator.yaml` e gerar o arquivo do pacote usando o `kustomization.yaml`.

  a. Crie o `kustomization.yaml` usando o seguinte comando onde *<bundle.yaml>* está `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` baseado na sua versão do Kubernetes.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. Compile o pacote usando o seguinte comando onde *<bundle.yaml>* está `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` baseado na sua versão do Kubernetes.

  ```
  kubectl kustomize deploy/ > deploy/<bundle.yaml>
  ```

Passos

1. Crie os recursos e implante o operador:

```
kubectl create -f deploy/<bundle.yaml>
```

2. Verifique se o operador, a implantação e as replicasets foram criados.

```
kubectl get all -n <operator-namespace>
```

Deve haver apenas **uma instância** do operador em um cluster do Kubernetes. Não crie várias implantações do operador Trident.

Passo 5: Atualize a localização do registo de imagens no `TridentOrchestrator`

O "Imagens de Trident e CSI" pode estar localizado em um Registro ou Registros diferentes, mas todas as imagens CSI devem estar localizadas no mesmo Registro. Atualize `deploy/crds/tridentorchestrator_cr.yaml` para adicionar as especificações de localização adicionais com base na configuração do seu registo.

Imagens em um Registro

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:24.02"
tridentImage: "<your-registry>/trident:24.02.0"
```

Imagens em diferentes registos

Você deve anexar `sig-storage` ao `imageRegistry` para usar diferentes locais de Registro.

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:24.02"
tridentImage: "<your-registry>/netapp/trident:24.02.0"
```

Passo 6: Crie o `TridentOrchestrator` e instale o Trident

Agora você pode criar e instalar o `TridentOrchestrator` Astra Trident. Opcionalmente, você pode usar ainda mais "Personalize a instalação do Trident"os atributos na `TridentOrchestrator` especificação. O exemplo a seguir mostra uma instalação onde as imagens Trident e CSI estão localizadas em diferentes Registros.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:24.02
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:24.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:24.02
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:24.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v24.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verifique a instalação

Existem várias maneiras de verificar sua instalação.

`TridentOrchestrator`Usando o status

O status de `TridentOrchestrator` indica se a instalação foi bem-sucedida e exibe a versão do Trident instalado. Durante a instalação, o status das `TridentOrchestrator` alterações de `Installing` para `Installed`. Se você observar o `Failed` status e o operador não conseguir recuperar sozinho, "verifique os logs".

Estado	Descrição
A instalar	O operador está instalando o Astra Trident usando este `TridentOrchestrator` CR.
Instalado	O Astra Trident foi instalado com sucesso.
Desinstalação	O operador está desinstalando o Astra Trident, `spec.uninstall=true` porque .
Desinstalado	O Astra Trident foi desinstalado.
Falha	O operador não pôde instalar, corrigir, atualizar ou desinstalar o Astra Trident; o operador tentará recuperar automaticamente deste estado. Se este estado persistir, será necessário resolver o problema.
A atualizar	O operador está atualizando uma instalação existente.
Erro	O `TridentOrchestrator` não é utilizado. Outro já existe.

Usando o status de criação do pod

Você pode confirmar se a instalação do Astra Trident foi concluída analisando o status dos pods criados:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw   6/6     Running   0
1m
trident-node-linux-mr6zc              2/2     Running   0
1m
trident-node-linux-xrp7w              2/2     Running   0
1m
trident-node-linux-zh2jt              2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0
3m
```

Utilização `tridentctl`

Você pode usar `tridentctl` para verificar a versão do Astra Trident instalada.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+
```

Implantar operador Trident usando Helm (modo padrão)

Você pode implantar o operador Trident e instalar o Astra Trident usando o Helm. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra Trident não são armazenadas em um Registro privado. Se tiver um registo de imagens privado, utilize o "processo para implantação off-line".

Informações críticas sobre o Astra Trident 24,02

Você deve ler as seguintes informações críticas sobre o Astra Trident.

** informações essenciais sobre o Astra Trident **

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante o operador Trident e instale o Astra Trident usando o Helm

Usando o Trident"Carta do leme", você pode implantar o operador Trident e instalar o Trident em uma etapa.

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Além do "pré-requisitos de implantação" que você precisa"Helm versão 3".

Passos

1. Adicione o repositório Astra Trident Helm:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Use `helm install` e especifique um nome para sua implantação, como no exemplo a seguir, onde `100.2402.0` está a versão do Astra Trident que você está instalando.

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--create-namespace --namespace <trident-namespace>
```

Se você já criou um namespace para Trident, o `--create-namespace` parâmetro não criará um namespace adicional.

Você pode usar `helm list` para revisar detalhes de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de revisão.

Passe os dados de configuração durante a instalação

Há duas maneiras de passar dados de configuração durante a instalação:

Opção	Descrição
`--values` (ou `-f`)	Especifique um arquivo YAML com substituições. Isso pode ser especificado várias vezes e o arquivo mais à direita terá precedência.
`--set`	Especifique substituições na linha de comando.

Por exemplo, para alterar o valor padrão `debug` do , execute o seguinte `--set` comando onde `100.2402.0` está a versão do Astra Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--create-namespace --namespace trident --set tridentDebug=true
```

Opções de configuração

Esta tabela e o `values.yaml` arquivo, que faz parte do gráfico Helm, fornecem a lista de chaves e seus valores padrão.

Opção	Descrição	Padrão
`nodeSelector`	Etiquetas de nó para atribuição de pod	
`podAnnotations`	Anotações do pod	
`deploymentAnnotations`	Anotações de implantação	
`tolerations`	Tolerâncias para atribuição de pod	

Opção	Descrição	Padrão
`affinity`	Afinidade para atribuição de pod	
`tridentControllerPluginNodeSelector`	Seletores de nós adicionais para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentControllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentNodePluginNodeSelector`	Seletores de nós adicionais para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentNodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`imageRegistry`	Identifica o registo para as `trident-operator`, `trident`e outras imagens. Deixe vazio para aceitar o padrão.	`""`
`imagePullPolicy`	Define a política de recebimento de imagens para o `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Define os segredos de extração da imagem para as `trident-operator`, `trident` e outras imagens.	
`kubeletDir`	Permite substituir a localização do host do estado interno do kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Permite que o nível de log do operador Trident seja definido como: `trace`, , `debug`, `info` `warn` , , `error` Ou `fatal`.	`"info"`
`operatorDebug`	Permite que o nível de log do operador Trident seja definido como debug.	`true`
`operatorImage`	Permite a substituição completa da imagem para `trident-operator`.	`""`
`operatorImageTag`	Permite substituir a etiqueta da `trident-operator` imagem.	`""`
`tridentIPv6`	Permite que o Astra Trident funcione em IPv6 clusters.	`false`
`tridentK8sTimeout`	Substitui o tempo limite padrão de 30 segundos para a maioria das operações da API do Kubernetes (se não for zero, em segundos).	`0`
`tridentHttpRequestTimeout`	Substitui o tempo limite padrão de 90 segundos para as solicitações HTTP, `0s` sendo uma duração infinita para o tempo limite. Valores negativos não são permitidos.	`"90s"`
`tridentSilenceAutosupport`	Permite desativar os relatórios periódicos do Astra Trident AutoSupport.	`false`

Opção	Descrição	Padrão
`tridentAutosupportImageTag`	Permite substituir a tag da imagem para o contentor Astra Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Permite que o Astra Trident AutoSupport Container ligue para casa por meio de um proxy HTTP.	`""`
`tridentLogFormat`	Define o formato de registo Astra Trident (`text`ou `json`).	`"text"`
`tridentDisableAuditLog`	Desativa o registrador de auditoria Astra Trident.	`true`
`tridentLogLevel`	Permite que o nível de log do Astra Trident seja definido como: `trace`, `debug`, , `info` `warn` , , `error` `fatal` Ou .	`"info"`
`tridentDebug`	Permite que o nível de log do Astra Trident seja definido como `debug`.	`false`
`tridentLogWorkflows`	Permite que fluxos de trabalho específicos do Astra Trident sejam ativados para registo de rastreio ou supressão de registos.	`""`
`tridentLogLayers`	Permite que camadas específicas do Astra Trident sejam ativadas para registo de rastreio ou supressão de registos.	`""`
`tridentImage`	Permite a substituição completa da imagem para Astra Trident.	`""`
`tridentImageTag`	Permite substituir a tag da imagem para Astra Trident.	`""`
`tridentProbePort`	Permite substituir a porta padrão usada para sondas de disponibilidade/prontidão do Kubernetes.	`""`
`windows`	Permite que o Astra Trident seja instalado no nó de trabalho do Windows.	`false`
`enableForceDetach`	Permite ativar a função forçar desanexar.	`false`
`excludePodSecurityPolicy`	Exclui a criação da diretiva de segurança do pod do operador.	`false`
`cloudProvider`	Defina como `"Azure"` quando utilizar identidades geridas ou uma identidade de nuvem num cluster AKS. Defina como "AWS" ao usar uma identidade de nuvem em um cluster EKS.	`""`
`cloudIdentity`	Defina como identidade da carga de trabalho ("azure.Workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxxxxxxx") ao usar identidade da nuvem em um cluster AKS. Defina como função do AWS IAM ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/astratrident-role'") ao usar a identidade da nuvem em um cluster do EKS.	`""`
`iscsiSelfHealingInterval`	O intervalo no qual a auto-recuperação iSCSI é invocada.	`5m0s`

Opção	Descrição	Padrão
`iscsiSelfHealingWaitTime`	A duração após a qual a auto-recuperação iSCSI inicia uma tentativa de resolver uma sessão obsoleta executando um logout e login subsequente.	`7m0s`

Compreensão dos pods dos nós e dos pods do controlador

O Astra Trident é executado como um único pod de controlador, além de um pod de nó em cada nó de trabalho no cluster. O pod de nó deve estar em execução em qualquer host onde você queira potencialmente montar um volume Astra Trident.

Kubernetes "seletores de nós" e "tolerações e taints"são usados para restringir um pod a ser executado em um nó específico ou preferencial. Usando o ControllerPlugin e `NodePlugin`o , você pode especificar restrições e substituições.

- O plugin controlador lida com o provisionamento e gerenciamento de volume, como snapshots e redimensionamento.
- O plug-in do nó manipula a conexão do armazenamento ao nó.

Implantar operador Trident usando Helm (modo off-line)

Você pode implantar o operador Trident e instalar o Astra Trident usando o Helm. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra Trident são armazenadas em um Registro privado. Se não tiver um registo de imagens privado, utilize o "processo para implantação padrão".

Informações críticas sobre o Astra Trident 24,02

Você deve ler as seguintes informações críticas sobre o Astra Trident.

informações essenciais sobre o Astra Trident

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante o operador Trident e instale o Astra Trident usando o Helm

Usando o Trident"Carta do leme", você pode implantar o operador Trident e instalar o Trident em uma etapa.

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Além do "pré-requisitos de implantação" que você precisa"Helm versão 3".

Passos

1. Adicione o repositório Astra Trident Helm:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Use `helm install` e especifique um nome para a localização do Registro de imagens e implantação. O "Imagens de Trident e CSI" pode estar localizado em um Registro ou Registros diferentes, mas todas as imagens CSI devem estar localizadas no mesmo Registro. Nos exemplos `100.2402.0`, é a versão do Astra Trident que você está instalando.

Imagens em um Registro

```
helm install <name> netapp-trident/trident-operator --version
100.2402.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

Imagens em diferentes registos

Você deve anexar `sig-storage` ao `imageRegistry` para usar diferentes locais de Registro.

```
helm install <name> netapp-trident/trident-operator --version
100.2402.0 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:24.02 --set tridentImage=<your-
registry>/netapp/trident:24.02.0 --create-namespace --namespace
<trident-namespace>
```

Se você já criou um namespace para Trident, o `--create-namespace` parâmetro não criará um namespace adicional.

Você pode usar `helm list` para revisar detalhes de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de revisão.

Passe os dados de configuração durante a instalação

Há duas maneiras de passar dados de configuração durante a instalação:

Opção	Descrição
`--values` (ou `-f`)	Especifique um arquivo YAML com substituições. Isso pode ser especificado várias vezes e o arquivo mais à direita terá precedência.

Opção	Descrição
`--set`	Especifique substituições na linha de comando.

Por exemplo, para alterar o valor padrão `debug` do , execute o seguinte `--set` comando onde `100.2402.0` está a versão do Astra Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--create-namespace --namespace trident --set tridentDebug=true
```

Opções de configuração

Esta tabela e o `values.yaml` arquivo, que faz parte do gráfico Helm, fornecem a lista de chaves e seus valores padrão.

Opção	Descrição	Padrão
`nodeSelector`	Etiquetas de nó para atribuição de pod	
`podAnnotations`	Anotações do pod	
`deploymentAnnotations`	Anotações de implantação	
`tolerations`	Tolerâncias para atribuição de pod	
`affinity`	Afinidade para atribuição de pod	
`tridentControllerPluginNodeSelector`	Seletores de nós adicionais para pods. "Compreensão dos pods dos nós e dos pods do controlador"Consulte para obter detalhes.	
`tridentControllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. "Compreensão dos pods dos nós e dos pods do controlador"Consulte para obter detalhes.	
`tridentNodePluginNodeSelector`	Seletores de nós adicionais para pods. "Compreensão dos pods dos nós e dos pods do controlador"Consulte para obter detalhes.	
`tridentNodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. "Compreensão dos pods dos nós e dos pods do controlador"Consulte para obter detalhes.	

Opção	Descrição	Padrão
`imageRegistry`	Identifica o registo para as `trident-operator`, `trident`e outras imagens. Deixe vazio para aceitar o padrão.	""
`imagePullPolicy`	Define a política de recebimento de imagens para o `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Define os segredos de extração da imagem para as `trident-operator`, `trident` e outras imagens.	
`kubeletDir`	Permite substituir a localização do host do estado interno do kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Permite que o nível de log do operador Trident seja definido como: `trace`, , `debug`, `info` `warn` , , `error` Ou `fatal`.	`"info"`
`operatorDebug`	Permite que o nível de log do operador Trident seja definido como debug.	`true`
`operatorImage`	Permite a substituição completa da imagem para `trident-operator`.	""
`operatorImageTag`	Permite substituir a etiqueta da `trident-operator` imagem.	""
`tridentIPv6`	Permite que o Astra Trident funcione em IPv6 clusters.	`false`
`tridentK8sTimeout`	Substitui o tempo limite padrão de 30 segundos para a maioria das operações da API do Kubernetes (se não for zero, em segundos).	`0`
`tridentHttpRequestTimeout`	Substitui o tempo limite padrão de 90 segundos para as solicitações HTTP, `0s` sendo uma duração infinita para o tempo limite. Valores negativos não são permitidos.	`"90s"`
`tridentSilenceAutosupport`	Permite desativar os relatórios periódicos do Astra Trident AutoSupport.	`false`
`tridentAutosupportImageTag`	Permite substituir a tag da imagem para o contentor Astra Trident AutoSupport.	`<version>`

Opção	Descrição	Padrão
`tridentAutosupportProxy`	Permite que o Astra Trident AutoSupport Container ligue para casa por meio de um proxy HTTP.	""
`tridentLogFormat`	Define o formato de registo Astra Trident (`text`ou `json`).	`"text"`
`tridentDisableAuditLog`	Desativa o registrador de auditoria Astra Trident.	`true`
`tridentLogLevel`	Permite que o nível de log do Astra Trident seja definido como: `trace`, `debug`, , `info` `warn` , , `error` `fatal` Ou .	`"info"`
`tridentDebug`	Permite que o nível de log do Astra Trident seja definido como `debug`.	`false`
`tridentLogWorkflows`	Permite que fluxos de trabalho específicos do Astra Trident sejam ativados para registo de rastreio ou supressão de registos.	""
`tridentLogLayers`	Permite que camadas específicas do Astra Trident sejam ativadas para registo de rastreio ou supressão de registos.	""
`tridentImage`	Permite a substituição completa da imagem para Astra Trident.	""
`tridentImageTag`	Permite substituir a tag da imagem para Astra Trident.	""
`tridentProbePort`	Permite substituir a porta padrão usada para sondas de disponibilidade/prontidão do Kubernetes.	""
`windows`	Permite que o Astra Trident seja instalado no nó de trabalho do Windows.	`false`
`enableForceDetach`	Permite ativar a função forçar desanexar.	`false`
`excludePodSecurityPolicy`	Exclui a criação da diretiva de segurança do pod do operador.	`false`

Personalizar a instalação do operador Trident

O operador Trident permite personalizar a instalação do Astra Trident usando os atributos da `TridentOrchestrator` especificação. Se você quiser personalizar a instalação além do que `TridentOrchestrator` os argumentos permitem, considere usar `tridentctl` para gerar manifestos YAML personalizados para modificar conforme necessário.

Compreensão dos pods dos nós e dos pods do controlador

O Astra Trident é executado como um único pod de controlador, além de um pod de nó em cada nó de trabalho no cluster. O pod de nó deve estar em execução em qualquer host onde você queira potencialmente montar um volume Astra Trident.

Kubernetes "seletores de nós" e "tolerações e taints"são usados para restringir um pod a ser executado em um nó específico ou preferencial. Usando o ControllerPlugin e `NodePlugin`o , você pode especificar restrições e substituições.

- O plugin controlador lida com o provisionamento e gerenciamento de volume, como snapshots e redimensionamento.
- O plug-in do nó manipula a conexão do armazenamento ao nó.

Opções de configuração

> `spec.namespace` É especificado em `TridentOrchestrator` para indicar o namespace onde o Astra Trident está instalado. Este parâmetro **não pode ser atualizado após a instalação do Astra Trident**. Tentar fazê-lo faz com que o `TridentOrchestrator` status mude para `Failed`. O Astra Trident não se destina a ser migrado entre namespaces.

Esta tabela detalha `TridentOrchestrator` atributos.

Parâmetro	Descrição	Padrão
`namespace`	Namespace para instalar Astra Trident em	`"default"`
`debug`	Habilite a depuração para o Astra Trident	`false`
`enableForceDetach`	`ontap-san` e `ontap-san-economy` apenas. Funciona com o Kubernetes Non-Graceful Node Shutdown (NGNS) para conceder aos administradores de cluster a capacidade de migrar com segurança cargas de trabalho com volumes montados para novos nós caso um nó não seja saudável.	`false`
`windows`	A configuração para `true` permite a instalação em nós de trabalho do Windows.	`false`
`cloudProvider`	Defina como `"Azure"` quando utilizar identidades geridas ou uma identidade de nuvem num cluster AKS. Defina como "AWS" ao usar uma identidade de nuvem em um cluster EKS.	`""`
`cloudIdentity`	Defina como identidade da carga de trabalho ("azure.Workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxxxxxxx") ao usar identidade da nuvem em um cluster AKS. Defina como função do AWS IAM ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/astratrident-role") ao usar a identidade da nuvem em um cluster do EKS.	`""`
`IPv6`	Instalar o Astra Trident em IPv6	falso
`k8sTimeout`	Tempo limite para operações do Kubernetes	`30sec`

Parâmetro	Descrição	Padrão
`silenceAutosupport`	Não envie pacotes AutoSupport para o NetApp automaticamente	`false`
`autosupportImage`	A imagem do recipiente para a telemetria AutoSupport	`"netapp/trident-autosupport:24.02"`
`autosupportProxy`	O endereço/porta de um proxy para o envio de telemetria AutoSupport	`"http://proxy.example.com:8888"`
`uninstall`	Um sinalizador usado para desinstalar o Astra Trident	`false`
`logFormat`	Formato de log Astra Trident a ser usado [text,json]	`"text"`
`tridentImage`	Imagem Astra Trident a instalar	`"netapp/trident:24.02"`
`imageRegistry`	Caminho para o Registro interno, do formato `<registry FQDN>[:port][/subpath]`	`"k8s.gcr.io/sig-storage"` (Kubernetes mais de 1,19) ou `"quay.io/k8scsi"`
`kubeletDir`	Caminho para o diretório kubelet no host	`"/var/lib/kubelet"`
`wipeout`	Uma lista de recursos a serem excluídos para realizar uma remoção completa do Astra Trident	
`imagePullSecrets`	Segredos para extrair imagens de um Registro interno	
`imagePullPolicy`	Define a política de recebimento de imagens para o operador Trident. Os valores válidos são: `Always` Para sempre puxar a imagem. `IfNotPresent` para puxar a imagem apenas se ela ainda não existir no nó. `Never` para nunca puxar a imagem.	`IfNotPresent`
`controllerPluginNodeSelector`	Seletores de nós adicionais para pods. Segue o mesmo formato que `pod.spec.nodeSelector`.	Sem padrão; opcional
`controllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que `pod.spec.Tolerations`.	Sem padrão; opcional
`nodePluginNodeSelector`	Seletores de nós adicionais para pods. Segue o mesmo formato que `pod.spec.nodeSelector`.	Sem padrão; opcional
`nodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que `pod.spec.Tolerations`.	Sem padrão; opcional

Para obter mais informações sobre a formatação dos parâmetros do pod, "Atribuindo pods a nós"consulte .

Detalhes sobre Force Detach

O Force Detach está disponível apenas para `ontap-san` e. `ontap-san-economy` Antes de ativar a desconexão forçada, o desligamento do nó (NGNS) não gracioso deve ser ativado no cluster do Kubernetes. Para obter mais informações, "Kubernetes: Desligamento do nó não gracioso"consulte .

Como o Astra Trident conta COM NGNS do Kubernetes, não `out-of-service` remova as taints de um nó que não seja saudável até que todos os workloads não toleráveis sejam reprogramados. Aplicar ou remover a taint de forma imprudente pode comprometer a proteção de dados no back-end.

Quando o administrador do cluster do Kubernetes tiver aplicado a `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` alteração ao nó e `enableForceDetach` estiver definido como `true`, o Astra Trident determinará o status do nó e:

1. Cessar o acesso de e/S de back-end para volumes montados nesse nó.
2. Marque o objeto nó Astra Trident como `dirty` (não é seguro para novas publicações).

O controlador Trident rejeitará novas solicitações de volume de publicação até que o nó seja requalificado (depois de ter sido marcado como `dirty`) pelo pod de nó do Trident. Quaisquer cargas de trabalho agendadas com um PVC montado (mesmo depois que o nó do cluster estiver pronto e saudável) não serão aceitas até que o Astra Trident possa verificar o nó `clean` (seguro para novas publicações).

Quando a integridade do nó é restaurada e a taint é removida, o Astra Trident:

1. Identifique e limpe caminhos publicados obsoletos no nó.
2. Se o nó estiver em um `cleanable` estado (a alteração fora de serviço foi removida e o nó está `Ready` no estado) e todos os caminhos obsoletos e publicados estiverem limpos, o Astra Trident reajustará o nó como `clean` e permitirá novos volumes publicados no nó.

Exemplos de configurações

Você pode usar os atributos em Opções de configuração ao definir `TridentOrchestrator` para personalizar sua instalação.

Configuração personalizada básica

Este é um exemplo para uma instalação personalizada básica.

```
cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

Seletores de nós

Este exemplo instala o Astra Trident com seletores de nós.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

Nós de trabalho do Windows

Este exemplo instala o Astra Trident em um nó de trabalho do Windows.

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

Identidades gerenciadas em um cluster AKS

Este exemplo instala o Astra Trident para habilitar identidades gerenciadas em um cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

Identidade de nuvem em um cluster AKS

Este exemplo instala o Astra Trident para uso com uma identidade de nuvem em um cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx'
```

Identidade de nuvem em um cluster EKS

Este exemplo instala o Astra Trident para uso com uma identidade de nuvem em um cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/astratrident-role'"
```

Instale usando o tridentctl

Instale usando o tridentctl

Você pode instalar o Astra Trident usando `tridentctl`o . Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Astra Trident são armazenadas em um Registro privado ou não. Para personalizar sua `tridentctl` implantação, "Personalizar a implantação do tridentctl"consulte .

Informações críticas sobre o Astra Trident 24,02

Você deve ler as seguintes informações críticas sobre o Astra Trident.

informações essenciais sobre o Astra Trident

- O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.
- O Astra Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Instalar o Astra Trident usando `tridentctl`

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Antes de iniciar a instalação, faça login no host Linux e verifique se ele está gerenciando um trabalho e "Cluster compatível com Kubernetes"se você tem o Privileges necessário.

> Com OpenShift, use `oc` em vez de `kubectl` em todos os exemplos que se seguem, e faça login como **system:admin** primeiro executando `oc login -u system:admin` ou `oc login -u kube-admin`.

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique o Privileges do administrador do cluster:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verifique se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Passo 1: Baixe o pacote de instalação do Trident

O pacote de instalação do Astra Trident cria um pod Trident, configura os objetos CRD que são usados para manter seu estado e inicializa os sidecars CSI para executar ações como provisionar e anexar volumes aos hosts de cluster. Baixe e extraia a versão mais recente do instalador do Trident a partir "A seção *assets* no GitHub"do . Atualize _ cliente Trident-installer-XX.XX.X.tar.gz>_ no exemplo com a versão selecionada do Astra Trident.

```
wget https://github.com/NetApp/trident/releases/download/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.tar.gz
cd trident-installer
```

Etapa 2: Instale o Astra Trident

Instale o Astra Trident no namespace desejado executando o `tridentctl install` comando. Você pode adicionar argumentos adicionais para especificar a localização do Registro da imagem.

Modo padrão

```
./tridentctl install -n trident
```

Imagens em um Registro

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:24.02 --trident
-image <your-registry>/trident:24.02.0
```

Imagens em diferentes registos

Você deve anexar `sig-storage` ao `imageRegistry` para usar diferentes locais de Registro.

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:24.02 --trident-image <your-
registry>/netapp/trident:24.02.0
```

Seu status de instalação deve ser parecido com isso.

```
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=24.02.0
INFO Trident installation succeeded.
....
```

Verifique a instalação

Você pode verificar sua instalação usando o status de criação do pod ou `tridentctl`.

Usando o status de criação do pod

Você pode confirmar se a instalação do Astra Trident foi concluída analisando o status dos pods criados:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     Running   0          5m29s
```

Se o instalador não for concluído com êxito ou `trident-controller-<generated id>` (`trident-csi-<generated id>` em versões anteriores a 23,01) não tiver um status **Running**, a plataforma não foi instalada. Use `-d` para "ative o modo de depuração" e solucione o problema.

Utilização `tridentctl`

Você pode usar `tridentctl` para verificar a versão do Astra Trident instalada.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+
```

Exemplos de configurações

Os exemplos a seguir fornecem exemplos de configurações para a instalação do Astra Trident usando `tridentctl`o .

Nós do Windows

Para permitir que o Astra Trident seja executado em nós do Windows:

```
tridentctl install --windows -n trident
```

Força a soltar

Para obter mais informações sobre a força desapegada, "Personalizar a instalação do operador Trident"consulte .

```
tridentctl install --enable-force-detach=true -n trident
```

Personalize a instalação do tridentctl

Você pode usar o instalador do Astra Trident para personalizar a instalação.

Saiba mais sobre o instalador

O instalador do Astra Trident permite personalizar atributos. Por exemplo, se você tiver copiado a imagem Trident para um repositório privado, poderá especificar o nome da imagem `--trident-image` usando o . Se você copiou a imagem do Trident, bem como as imagens do sidecar do CSI necessárias para um repositório privado, pode ser preferível especificar a localização desse repositório usando o `--image-registry` switch, que assume o formulário `<registry FQDN>[:port]`.

Se você estiver usando uma distribuição do Kubernetes, onde `kubelet` mantém seus dados em um caminho diferente do habitual `/var/lib/kubelet`, você poderá especificar o caminho alternativo usando `--kubelet-dir`o .

Se você precisar personalizar a instalação além do que os argumentos do instalador permitem, você também pode personalizar os arquivos de implantação. Usando o `--generate-custom-yaml` parâmetro cria os seguintes arquivos YAML no diretório do instalador `setup`:

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`
- `trident-resourcequota.yaml`

Depois de gerar esses arquivos, você pode modificá-los de acordo com suas necessidades e, em seguida, usá-los `--use-custom-yaml` para instalar sua implantação personalizada.

```
./tridentctl install -n trident --use-custom-yaml
```

Use o Astra Trident

Prepare o nó de trabalho

Todos os nós de trabalho no cluster do Kubernetes precisam ser capazes de montar os volumes provisionados para os pods. Para preparar os nós de trabalho, é necessário instalar ferramentas NFS, iSCSI ou NVMe/TCP com base na seleção de driver.

Selecionar as ferramentas certas

Se você estiver usando uma combinação de drivers, você deve instalar todas as ferramentas necessárias para seus drivers. Versões recentes do RedHat CoreOS têm as ferramentas instaladas por padrão.

Ferramentas NFS

"Instalar as ferramentas NFS" se estiver a utilizar: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, , , `azure-netapp-files` `gcp-cvs` .

Ferramentas iSCSI

"Instale as ferramentas iSCSI" se estiver a utilizar: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

Ferramentas NVMe

"Instalar as ferramentas do NVMe" Se você estiver usando `ontap-san` o protocolo NVMe (não-volátil Memory Express) em TCP (NVMe/TCP).

> Recomendamos o ONTAP 9.12 ou posterior para NVMe/TCP.

Detecção de serviço de nós

O Astra Trident tenta detetar automaticamente se o nó pode executar serviços iSCSI ou NFS.

> A descoberta de serviço de nó identifica os serviços descobertos, mas não garante que os serviços estejam configurados corretamente. Por outro lado, a ausência de um serviço descoberto não garante que a montagem de volume falhe.

Rever eventos

O Astra Trident cria eventos para o nó a fim de identificar os serviços descobertos. Para rever estes eventos, execute:

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

Reveja os serviços descobertos

O Astra Trident identifica serviços habilitados para cada nó no nó CR da Trident. Para visualizar os serviços descobertos, execute:

```
tridentctl get node -o wide -n <Trident namespace>
```

Volumes NFS

Instale as ferramentas NFS usando os comandos do seu sistema operacional. Certifique-se de que o serviço NFS seja iniciado durante o tempo de inicialização.

RHEL 8 MAIS

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

Reinicie seus nós de trabalho após instalar as ferramentas NFS para evitar falhas ao anexar volumes a contêineres.

Volumes iSCSI

O Astra Trident pode estabelecer automaticamente uma sessão iSCSI, digitalizar LUNs e descobrir dispositivos multipath, formatá-los e montá-los em um pod.

Recursos de autorrecuperação iSCSI

Para sistemas ONTAP, o Astra Trident executa autorrecuperação iSCSI a cada cinco minutos para:

1. **Identifique** o estado de sessão iSCSI desejado e o estado atual da sessão iSCSI.
2. **Compare** o estado desejado com o estado atual para identificar as reparações necessárias. O Astra Trident determina as prioridades de reparação e quando efetuar as reparações.
3. **Efetuar reparações** necessárias para repor o estado atual da sessão iSCSI para o estado de sessão iSCSI pretendido.

Logs de atividade de auto-cura estão localizados no `trident-main` recipiente no respetivo pod Daemonset. Para visualizar logs, você deve ter definido `debug` como "true" durante a instalação do Astra Trident.

As funcionalidades de autorrecuperação iSCSI do Astra Trident ajudam a impedir:

- Sessões iSCSI obsoletas ou não saudáveis que podem ocorrer após um problema de conetividade de rede. No caso de uma sessão obsoleta, o Astra Trident aguarda sete minutos antes de sair para restabelecer a conexão com um portal.

Por exemplo, se os segredos CHAP foram girados no controlador de armazenamento e a rede perder a conetividade, os segredos CHAP antigos (*stale*) podem persistir. A auto-cura pode reconhecer isso e restabelecer automaticamente a sessão para aplicar os segredos CHAP atualizados.

- Sessões iSCSI em falta
- LUNs em falta

Instale as ferramentas iSCSI

Instale as ferramentas iSCSI utilizando os comandos do seu sistema operativo.

Antes de começar

- Cada nó no cluster do Kubernetes precisa ter uma IQN exclusiva. **Este é um pré-requisito necessário**.
- Se estiver usando RHCOS versão 4,5 ou posterior, ou outra distribuição Linux compatível com RHEL, com o `solidfire-san` driver e o Element OS 12,5 ou anterior, verifique se o algoritmo de autenticação CHAP está definido como MD5 em `/etc/iscsi/iscsid.conf`. algoritmos CHAP compatíveis com FIPS seguros SHA1, SHA-256 e SHA3-256 estão disponíveis com o elemento 12,7.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- Ao usar nós de trabalho que executam RHEL/RedHat CoreOS com iSCSI PVs, especifique a `discard` mountOption no StorageClass para executar a recuperação de espaço em linha. Consulte a "Documentação da RedHat".

RHEL 8 MAIS

1. Instale os seguintes pacotes de sistema:

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-
mapper-multipath
```

2. Verifique se a versão iscsi-iniciador-utils é 6,2.0,874-2.el7 ou posterior:

```
rpm -q iscsi-initiator-utils
```

3. Definir a digitalização para manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ativar multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de `etc/multipath.conf` que contém `find_multipaths no defaults` em .

5. Certifique-se de que `iscsid` e `multipathd` estão a funcionar:

```
sudo systemctl enable --now iscsid multipathd
```

6. Ativar e iniciar `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Instale os seguintes pacotes de sistema:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Verifique se a versão Open-iscsi é 2,0.874-5ubuntu2.10 ou posterior (para bionic) ou 2,0.874-7.1ubuntu6.1 ou posterior (para focal):

```
dpkg -l open-iscsi
```

3. Definir a digitalização para manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ativar multipathing:

```
sudo tee /etc/multipath.conf <<-'EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de `etc/multipath.conf` que contém `find_multipaths no` `defaults` em .

5. Certifique-se de que `open-iscsi` e `multipath-tools` estão ativados e em execução:

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> Para o Ubuntu 18,04, você deve descobrir portas de destino com `iscsiadm` antes de iniciar `open-iscsi` o daemon iSCSI para iniciar. Em alternativa, pode modificar o `iscsi` serviço para iniciar `iscsid` automaticamente.

Configurar ou desativar a auto-recuperação iSCSI

Você pode configurar as seguintes configurações de autorrecuperação iSCSI Astra Trident para corrigir sessões obsoletas:

- **Intervalo de auto-recuperação iSCSI**: Determina a frequência na qual a auto-recuperação iSCSI é invocada (predefinição: 5 minutos). Você pode configurá-lo para executar com mais frequência definindo um número menor ou com menos frequência definindo um número maior.

Definir o intervalo de auto-recuperação iSCSI para 0 interrompe completamente a auto-recuperação iSCSI. Não recomendamos a desativação do iSCSI Self-healing; ele só deve ser desativado em certos cenários quando o iSCSI Self-healing não estiver funcionando como pretendido ou para fins de depuração.

- **Tempo de espera de auto-cura iSCSI**: Determina a duração de espera de auto-recuperação iSCSI antes de sair de uma sessão não saudável e tentar iniciar sessão novamente (predefinição: 7 minutos). Você pode configurá-lo para um número maior para que as sessões identificadas como não saudáveis tenham que esperar mais antes de serem desconetadas e, em seguida, uma tentativa é feita para fazer login novamente, ou um número menor para fazer logout e fazer login mais cedo.

Leme

Para configurar ou alterar as definições de recuperação automática iSCSI, passe os `iscsiSelfHealingInterval` parâmetros e `iscsiSelfHealingWaitTime` durante a instalação do leme ou atualização do leme.

O exemplo a seguir define o intervalo de auto-recuperação iSCSI para 3 minutos e o tempo de espera de auto-recuperação para 6 minutos:

```
helm install trident trident-operator-100.2402.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

Para configurar ou alterar as configurações de auto-recuperação iSCSI, passe os `iscsi-self-healing-interval` parâmetros e `iscsi-self-healing-wait-time` durante a instalação ou atualização do tridentctl.

O exemplo a seguir define o intervalo de auto-recuperação iSCSI para 3 minutos e o tempo de espera de auto-recuperação para 6 minutos:

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

Volumes NVMe/TCP

Instale as ferramentas NVMe usando os comandos do seu sistema operacional.

- O NVMe requer o RHEL 9 ou posterior.
- Se a versão do kernel do seu nó Kubernetes for muito antiga ou se o pacote NVMe não estiver disponível para a versão do kernel, talvez seja necessário atualizar a versão do kernel do nó para uma com o pacote NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Verifique a instalação

Após a instalação, verifique se cada nó no cluster do Kubernetes tem um NQN exclusivo usando o comando:

```
cat /etc/nvme/hostnqn
```

O Astra Trident modifica o `ctrl_device_tmo` valor para garantir que o NVMe não desista do caminho se ele cair. Não altere esta definição.

Configurar e gerenciar backends

Configurar backends

Um back-end define a relação entre o Astra Trident e um sistema de storage. Ele diz ao Astra Trident como se comunicar com esse sistema de storage e como o Astra Trident deve provisionar volumes a partir dele.

O Astra Trident oferece automaticamente pools de storage de back-ends que atendem aos requisitos definidos por uma classe de storage. Saiba como configurar o back-end para o seu sistema de armazenamento.

- "Configurar um back-end do Azure NetApp Files"
- "Configure um back-end do Cloud Volumes Service para o Google Cloud Platform"
- "Configurar um back-end NetApp HCI ou SolidFire"
- "Configurar um back-end com drivers nas ONTAP ou Cloud Volumes ONTAP"
- "Configure um back-end com drivers SAN ONTAP ou Cloud Volumes ONTAP"
- "Use o Astra Trident com o Amazon FSX para NetApp ONTAP"

Azure NetApp Files

Configurar um back-end do Azure NetApp Files

Você pode configurar o Azure NetApp Files como back-end para o Astra Trident. É possível anexar volumes NFS e SMB usando um back-end do Azure NetApp Files. O Astra Trident também oferece suporte ao gerenciamento de credenciais usando identidades gerenciadas para clusters do Azure Kubernetes Services (AKS).

Detalhes do driver Azure NetApp Files

O Astra Trident fornece os seguintes drivers de storage Azure NetApp Files para se comunicar com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMuy* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`azure-netapp-files`	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Considerações

- O serviço Azure NetApp Files não oferece suporte a volumes menores que 100 GB. O Astra Trident cria automaticamente volumes de 100 GiB se um volume menor for solicitado.
- O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows.

Identidades gerenciadas para AKS

O Astra Trident é compatível "identidades gerenciadas"com clusters do Azure Kubernetes Services. Para aproveitar o gerenciamento simplificado de credenciais oferecido por identidades gerenciadas, você deve ter:

- Um cluster do Kubernetes implantado usando AKS
- Identidades gerenciadas configuradas no cluster AKS kuquilla
- Astra Trident instalado que inclui o `cloudProvider` para especificar `"Azure"`.

Operador Trident

Para instalar o Astra Trident usando o operador Trident, edite `tridentorchestrator_cr.yaml` para definir `cloudProvider` como `"Azure"`. Por exemplo:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Leme

O exemplo a seguir instala conjuntos Astra Trident `cloudProvider` no Azure usando a variável de ambiente `$CP` :

```
helm install trident trident-operator-100.2402.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>dtridentctl</code>

O exemplo a seguir instala o Astra Trident e define o `cloudProvider` sinalizador como `Azure`:

```
tridentctl install --cloud-provider="Azure" -n trident
```

Identidade de nuvem para AKS

A identidade na nuvem permite que os pods do Kubernetes acessem recursos do Azure autenticando como uma identidade de workload em vez de fornecendo credenciais explícitas do Azure.

Para aproveitar a identidade da nuvem no Azure, você deve ter:

- Um cluster do Kubernetes implantado usando AKS
- Identidade da carga de trabalho e oidc-emitente configurados no cluster AKS Kubernetes
- Astra Trident instalado que inclui o `cloudProvider` para especificar `"Azure"` e `cloudIdentity` especificar a identidade da carga de trabalho

Operador Trident

Para instalar o Astra Trident usando o operador Trident, edite `tridentorchestrator_cr.yaml` para definir `cloudProvider` como `"Azure"` e defina `cloudIdentity` como `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Por exemplo:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  *cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx'*
```

Leme

Defina os valores para sinalizadores **provedor de nuvem (CP)** e **identidade de nuvem (IC)** usando as seguintes variáveis de ambiente:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

O exemplo a seguir instala o Astra Trident e define `cloudProvider` para o Azure usando a variável de ambiente `$CP` e define a `cloudIdentity` variável usando o ambiente `$CI` :

```
helm install trident trident-operator-100.2402.0.tgz --set
cloudProvider=$CP --set cloudIdentity=$CI
```

<code>dtridentctl</code>

Defina os valores para os sinalizadores **provedor de nuvem** e **identidade de nuvem** usando as seguintes variáveis de ambiente:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

O exemplo a seguir instala o Astra Trident e define o `cloud-provider` sinalizador como `$CP`, e `cloud-identity` como `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Prepare-se para configurar um back-end do Azure NetApp Files

Antes de configurar o back-end do Azure NetApp Files, você precisa garantir que os seguintes requisitos sejam atendidos.

Pré-requisitos para volumes NFS e SMB

Se você estiver usando o Azure NetApp Files pela primeira vez ou em um novo local, será necessária alguma configuração inicial para configurar o Azure NetApp Files e criar um volume NFS. Consulte a "Azure: Configure o Azure NetApp Files e crie um volume NFS".

Para configurar e usar um "Azure NetApp Files" back-end, você precisa do seguinte:

- `subscriptionID` `tenantID`, , `clientID`, , `location` E `clientSecret` são opcionais ao usar identidades gerenciadas em um cluster AKS.
- `tenantID` `clientID`, , E `clientSecret` são opcionais ao usar uma identidade de nuvem em um cluster AKS.

- Um pool de capacidade. "Microsoft: Crie um pool de capacidade para o Azure NetApp Files"Consulte a .
- Uma sub-rede delegada ao Azure NetApp Files. "Microsoft: Delegar uma sub-rede ao Azure NetApp Files"Consulte a .
- `subscriptionID` A partir de uma subscrição do Azure com o Azure NetApp Files ativado.
- `tenantID`, `clientID` E `clientSecret` de um "Registo da aplicação" no Azure ative Directory com permissões suficientes para o serviço Azure NetApp Files. O Registro de aplicativos deve usar:
  - A função proprietário ou Colaborador "Pré-definido pelo Azure".
  - A "Função de Colaborador personalizada"no nível da subscrição (`assignableScopes`) com as seguintes permissões limitadas apenas ao que o Astra Trident requer. Depois de criar a função personalizada"Atribua a função usando o portal do Azure", .

Função de colaborador personalizada

```
{
    "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
    "properties": {
        "roleName": "custom-role-with-limited-perms",
        "description": "custom role providing limited
permissions",
        "assignableScopes": [
            "/subscriptions/<subscription-id>"
        ],
        "permissions": [
            {
                "actions": [

"Microsoft.NetApp/netAppAccounts/capacityPools/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
                    "Microsoft.Network/virtualNetworks/read",

"Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
                    "Microsoft.Features/features/read",
                    "Microsoft.Features/operations/read",
                    "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
                ],
                "notActions": [],
                "dataActions": [],
                "notDataActions": []
            }
        ]
    }
}
```

- O Azure `location` que contém pelo menos um "sub-rede delegada". A partir do Trident 22,01, o `location` parâmetro é um campo obrigatório no nível superior do arquivo de configuração de back-end. Os valores de localização especificados em pools virtuais são ignorados.
- Para usar `Cloud Identity`o , obtenha o `client ID` de a "identidade gerenciada atribuída pelo usuário" e especifique esse ID no `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Requisitos adicionais para volumes SMB

Para criar um volume SMB, você deve ter:

- Ative Directory configurado e conetado ao Azure NetApp Files. "Microsoft: Crie e gerencie conexões do ative Directory para Azure NetApp Files"Consulte a .
- Um cluster do Kubernetes com um nó de controlador Linux e pelo menos um nó de trabalho do Windows que executa o Windows Server 2019. O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows.
- Pelo menos um segredo do Astra Trident que contém suas credenciais do ative Directory para que o Azure NetApp Files possa se autenticar no ative Directory. Para gerar segredo `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Um proxy CSI configurado como um serviço Windows. Para configurar um `csi-proxy`, "GitHub: CSI

Proxy"consulte ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes executados no Windows.

Exemplos e opções de configuração de back-end do Azure NetApp Files

Saiba mais sobre as opções de configuração de back-end NFS e SMB para Azure NetApp Files e reveja exemplos de configuração.

Opções de configuração de back-end

O Astra Trident usa sua configuração de back-end (sub-rede, rede virtual, nível de serviço e local) para criar volumes Azure NetApp Files em pools de capacidade disponíveis no local solicitado e que correspondam ao nível de serviço e à sub-rede solicitados.

O Astra Trident não é compatível com pools de capacidade de QoS manual.

Os backends Azure NetApp Files fornecem essas opções de configuração.

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	"ficheiros azure-NetApp"
`backendName`	Nome personalizado ou back-end de storage	Nome do condutor e carateres aleatórios
`subscriptionID`	O ID de assinatura da sua assinatura do Azure Opcional quando identidades gerenciadas está habilitado em um cluster AKS.	
`tenantID`	O ID do locatário de um Registo de aplicações Opcional quando identidades geridas ou identidade na nuvem são utilizadas num cluster AKS.	
`clientID`	A ID do cliente de um registo de aplicações opcional quando identidades geridas ou identidade na nuvem são utilizadas num cluster AKS.	
`clientSecret`	O segredo do cliente de um Registo de aplicações Opcional quando identidades geridas ou identidade na nuvem são utilizadas num cluster AKS.	
`serviceLevel`	Um de `Standard`, `Premium`, ou `Ultra`	"" (aleatório)

Parâmetro	Descrição	Padrão
`location`	Nome do local do Azure onde os novos volumes serão criados Opcional quando identidades gerenciadas estiverem ativadas em um cluster AKS.	
`resourceGroups`	Lista de grupos de recursos para filtragem de recursos descobertos	"[]" (sem filtro)
`netappAccounts`	Lista de contas do NetApp para filtragem de recursos descobertos	"[]" (sem filtro)
`capacityPools`	Lista de pools de capacidade para filtrar recursos descobertos	"[]" (sem filtro, aleatório)
`virtualNetwork`	Nome de uma rede virtual com uma sub-rede delegada	""
`subnet`	Nome de uma sub-rede delegada `Microsoft.Netapp/volumes`	""
`networkFeatures`	Conjunto de recursos VNet para um volume, pode ser `Basic` ou `Standard`. Os recursos de rede não estão disponíveis em todas as regiões e podem ter que ser ativados em uma assinatura. Especificar `networkFeatures` quando a funcionalidade não está ativada faz com que o provisionamento de volume falhe.	""
`nfsMountOptions`	Controle refinado das opções de montagem NFS. Ignorado para volumes SMB. Para montar volumes usando o NFS versão 4,1, inclua `nfsvers=4`na lista de opções de montagem delimitadas por vírgulas para escolher NFS v4,1. As opções de montagem definidas em uma definição de classe de armazenamento substituem as opções de montagem definidas na configuração de back-end.	"3"
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor	"" (não aplicado por padrão)

Parâmetro	Descrição	Padrão
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Exemplo, `\{"api": false, "method": true, "discovery": true}`. Não use isso a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. A configuração como null padrão para volumes NFS.	`nfs`

Para obter mais informações sobre recursos de rede, "Configurar recursos de rede para um volume Azure NetApp Files"consulte .

Permissões e recursos necessários

Se você receber um erro "sem pools de capacidade encontrados" ao criar um PVC, é provável que o Registro do aplicativo não tenha as permissões e recursos necessários (sub-rede, rede virtual, pool de capacidade) associados. Se a depuração estiver ativada, o Astra Trident registrará os recursos do Azure descobertos quando o back-end for criado. Verifique se uma função apropriada está sendo usada.

Os valores para `resourceGroups`, `netappAccounts`, `capacityPools`, , `virtualNetwork` e `subnet` podem ser especificados usando nomes curtos ou totalmente qualificados. Nomes totalmente qualificados são recomendados na maioria das situações, pois nomes curtos podem corresponder vários recursos com o mesmo nome.

Os `resourceGroups` valores , `netappAccounts`, e `capacityPools` são filtros que restringem o conjunto de recursos descobertos aos disponíveis para esse back-end de armazenamento e podem ser especificados em qualquer combinação. Nomes totalmente qualificados seguem este formato:

Tipo	Formato
Grupo de recursos	<resource group>
Conta NetApp	<resource group>/ cliente NetApp account>
Pool de capacidade	<resource group>/ cliente NetApp account>/<capacity pool>
Rede virtual	<resource group>/<virtual network>
Sub-rede	<resource group>/<virtual network>/<subnet>

Provisionamento de volume

Você pode controlar o provisionamento de volume padrão especificando as seguintes opções em uma seção especial do arquivo de configuração. Exemplos de configuraçõesConsulte para obter detalhes.

Parâmetro	Descrição	Padrão
`exportRule`	Regras de exportação para novos volumes. `exportRule` Deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4 ou sub-redes IPv4 na notação CIDR. Ignorado para volumes SMB.	"0,0.0,0/0"
`snapshotDir`	Controla a visibilidade do diretório .snapshot	"falso"
`size`	O tamanho padrão dos novos volumes	"100G"
`unixPermissions`	As permissões unix de novos volumes (4 dígitos octal). Ignorado para volumes SMB.	"" (recurso de pré-visualização, requer lista branca na assinatura)

Exemplos de configurações

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros padrão. Esta é a maneira mais fácil de definir um backend.

Configuração mínima

Esta é a configuração mínima absoluta de back-end. Com essa configuração, o Astra Trident descobre todas as suas contas NetApp, pools de capacidade e sub-redes delegadas ao Azure NetApp Files no local configurado e coloca novos volumes aleatoriamente em um desses pools e sub-redes. Como `nasType` é omitido, o `nfs` padrão se aplica e o back-end provisionará para volumes NFS.

Essa configuração é ideal quando você está apenas começando a usar o Azure NetApp Files e experimentando as coisas, mas na prática você vai querer fornecer um escopo adicional para os volumes provisionados.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
```

Identidades gerenciadas para AKS

Esta configuração de back-end omits , `subscriptionID` `tenantID`, `clientID`, e `clientSecret`, que são opcionais ao usar identidades gerenciadas.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

Identidade de nuvem para AKS

Essa configuração de back-end omits , `tenantID` `clientID`, e `clientSecret`, que são opcionais ao usar uma identidade de nuvem.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

Configuração específica de nível de serviço com filtros de pool de capacidade

Essa configuração de back-end coloca volumes no local do Azure `eastus` em um `Ultra` pool de capacidade. O Astra Trident descobre automaticamente todas as sub-redes delegadas ao Azure NetApp Files nesse local e coloca um novo volume em uma delas aleatoriamente.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
```

Configuração avançada

Essa configuração de back-end reduz ainda mais o escopo do posicionamento de volume para uma única sub-rede e também modifica alguns padrões de provisionamento de volume.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: 'true'
  size: 200Gi
  unixPermissions: '0777'
```

Configuração do pool virtual

Essa configuração de back-end define vários pools de storage em um único arquivo. Isso é útil quando você tem vários pools de capacidade com suporte a diferentes níveis de serviço e deseja criar classes de storage no Kubernetes que os representem. Rótulos de pool virtual foram usados para diferenciar os pools com base `performance` no .

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
- application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
- labels:
    performance: gold
  serviceLevel: Ultra
  capacityPools:
  - ultra-1
  - ultra-2
  networkFeatures: Standard
- labels:
    performance: silver
  serviceLevel: Premium
  capacityPools:
  - premium-1
- labels:
    performance: bronze
  serviceLevel: Standard
  capacityPools:
  - standard-1
  - standard-2
```

Definições da classe de armazenamento

As definições a seguir `StorageClass` referem-se aos pools de armazenamento acima.

Exemplos de definições usando `parameter.selector` campo

Usando `parameter.selector` você pode especificar para cada `StorageClass` pool virtual que é usado para hospedar um volume. O volume terá os aspetos definidos no pool escolhido.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze"
allowVolumeExpansion: true
```

Definições de exemplo para volumes SMB

Usando `nasType`, `node-stage-secret-name` e `node-stage-secret-namespace`, você pode especificar um volume SMB e fornecer as credenciais necessárias do ative Directory.

Configuração básica no namespace padrão

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Usando diferentes segredos por namespace

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

Usando diferentes segredos por volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb` Filtros para pools compatíveis com volumes SMB. `nasType: nfs` Ou `nasType: null` filtros para NFS Pools.

Crie o backend

Depois de criar o arquivo de configuração de back-end, execute o seguinte comando:

```
tridentctl create backend -f <backend-file>
```

Se a criação do backend falhar, algo está errado com a configuração do backend. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

Configure um back-end do Cloud Volumes Service para o Google Cloud

Saiba como configurar o NetApp Cloud Volumes Service para Google Cloud como back-end para sua instalação do Astra Trident usando as configurações de exemplo fornecidas.

Detalhes do driver do Google Cloud

O Astra Trident fornece ao `gcp-cvs` condutor a comunicação com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMuy* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`gcp-cvs`	NFS	Sistema de ficheiros	RWO, ROX, RWX, RWOP	`nfs`

Saiba mais sobre o suporte ao Astra Trident para Cloud Volumes Service para Google Cloud

O Astra Trident pode criar volumes Cloud Volumes Service em um de dois "tipos de serviço":

- **CVS-Performance**: O tipo de serviço padrão Astra Trident. Esse tipo de serviço otimizado para performance é mais adequado para workloads de produção que valorizam a performance. O tipo de serviço CVS-Performance é uma opção de hardware que suporta volumes com um tamanho mínimo de 100 GiB. Você pode escolher um dos "três níveis de serviço":
  - `standard`
  - `premium`
  - `extreme`
- **CVS**: O tipo de serviço CVS fornece alta disponibilidade por zonas com níveis de desempenho limitados a

moderados. O tipo de serviço CVS é uma opção de software que usa pools de armazenamento para dar suporte a volumes tão pequenos quanto 1 GiB. O pool de storage pode conter até 50 volumes em que todos os volumes compartilham a capacidade e a performance do pool. Você pode escolher um dos "dois níveis de serviço":

  - `standardsw`
  - `zoneredundantstandardsw`

O que você vai precisar

Para configurar e usar o "Cloud Volumes Service para Google Cloud" back-end, você precisa do seguinte:

- Uma conta do Google Cloud configurada com o NetApp Cloud Volumes Service
- Número do projeto da sua conta do Google Cloud
- Conta de serviço do Google Cloud com a `netappcloudvolumes.admin` função
- Arquivo de chave de API para sua conta Cloud Volumes Service

Opções de configuração de back-end

Cada back-end provisiona volumes em uma única região do Google Cloud. Para criar volumes em outras regiões, você pode definir backends adicionais.

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	"gcp-cvs"
`backendName`	Nome personalizado ou back-end de storage	Nome do driver e parte da chave da API
`storageClass`	Parâmetro opcional usado para especificar o tipo de serviço CVS. `software`Use para selecionar o tipo de serviço CVS. Caso contrário, o Astra Trident assume o tipo de serviço CVS-Performance (`hardware).	
`storagePools`	Apenas tipo de serviço CVS. Parâmetro opcional usado para especificar pools de armazenamento para criação de volume.	
`projectNumber`	Número do projeto da conta Google Cloud. O valor é encontrado na página inicial do portal do Google Cloud.	
`hostProjectNumber`	Necessário se estiver usando uma rede VPC compartilhada. Neste cenário, `projectNumber` é o projeto de serviço, e `hostProjectNumber` é o projeto host.	

Parâmetro	Descrição	Padrão
`apiRegion`	A região do Google Cloud onde o Astra Trident cria o Cloud Volumes Service volumes. Ao criar clusters de Kubernetes entre regiões, os volumes criados em um `apiRegion` podem ser usados em workloads programados em nós em várias regiões do Google Cloud. O tráfego entre regiões incorre em um custo adicional.	
`apiKey`	Chave de API para a conta de serviço do Google Cloud com a `netappcloudvolumes.admin` função. Ele inclui o conteúdo formatado em JSON do arquivo de chave privada de uma conta de serviço do Google Cloud (copiado literalmente no arquivo de configuração de back-end).	
`proxyURL`	URL do proxy se o servidor proxy for necessário para se conetar à conta CVS. O servidor proxy pode ser um proxy HTTP ou um proxy HTTPS. Para um proxy HTTPS, a validação do certificado é ignorada para permitir o uso de certificados autoassinados no servidor proxy. Os servidores proxy com autenticação ativada não são suportados.	
`nfsMountOptions`	Controle refinado das opções de montagem NFS.	"3"
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor.	"" (não aplicado por padrão)
`serviceLevel`	O nível de serviço CVS-Performance ou CVS para novos volumes. Os valores CVS-Performance são `standard`, `premium`, `extreme` ou . Os valores CVS são `standardsw` ou `zoneredundantstandardsw`.	O padrão CVS-Performance é "padrão". O padrão CVS é "standardsw".
`network`	Rede Google Cloud usada para Cloud Volumes Service volumes.	"predefinição"
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Exemplo, `\{"api":false, "method":true}`. Não use isso a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo

`allowedTopologies`	Para habilitar o acesso entre regiões, a definição do StorageClass para `allowedTopologies` deve incluir todas as regiões. Por exemplo:
	`- key: topology.kubernetes.io/region`
	`values:`
	`- us-east1`
	`- europe-west1`	

Opções de provisionamento de volume

Você pode controlar o provisionamento de volume padrão `defaults` na seção do arquivo de configuração.

Parâmetro	Descrição	Padrão
`exportRule`	As regras de exportação para novos volumes. Deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4 ou sub-redes IPv4 na notação CIDR.	"0,0.0,0/0"
`snapshotDir`	Acesso ao `.snapshot` diretório	"falso"
`snapshotReserve`	Porcentagem de volume reservado para snapshots	"" (aceitar o padrão CVS de 0)
`size`	O tamanho dos novos volumes. O mínimo de desempenho do CVS é de 100 GiB. CVS mínimo é de 1 GiB.	O tipo de serviço CVS-Performance é padrão para "100GiB". O tipo de serviço CVS não define um padrão, mas requer um mínimo de 1 GiB.

Exemplos de tipos de serviço CVS-Performance

Os exemplos a seguir fornecem exemplos de configurações para o tipo de serviço CVS-Performance.

Exemplo 1: Configuração mínima

Essa é a configuração mínima de back-end usando o tipo de serviço CVS-Performance padrão com o nível de serviço padrão.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
```

```
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
```

Exemplo 2: Configuração do nível de serviço

Este exemplo ilustra as opções de configuração de back-end, incluindo nível de serviço e padrões de volume.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
```

```
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```

Exemplo 3: Configuração de pool virtual

Este exemplo usa `storage` para configurar pools virtuais e os `StorageClasses` que se referem a eles. Definições de classe de armazenamentoConsulte para ver como as classes de armazenamento foram definidas.

Aqui, padrões específicos são definidos para todos os pools virtuais, que definem o `snapshotReserve` em 5% e o `exportRule` para 0,0.0,0/0. Os pools virtuais são definidos na `storage` seção. Cada pool virtual individual define seu próprio `serviceLevel`, e alguns pools substituem os valores padrão. Rótulos de pool virtual foram usados para diferenciar os pools com base em `performance` e `protection`.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
```

```
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
region: us-west2
storage:
- labels:
    performance: extreme
    protection: extra
  serviceLevel: extreme
  defaults:
    snapshotDir: 'true'
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
- labels:
    performance: extreme
    protection: standard
  serviceLevel: extreme
- labels:
    performance: premium
    protection: extra
  serviceLevel: premium
  defaults:
    snapshotDir: 'true'
    snapshotReserve: '10'
- labels:
    performance: premium
    protection: standard
  serviceLevel: premium
- labels:
    performance: standard
```

```
    serviceLevel: standard
```

Definições de classe de armazenamento

As seguintes definições do StorageClass se aplicam ao exemplo de configuração de pool virtual. Usando `parameters.selector`o , você pode especificar para cada StorageClass o pool virtual usado para hospedar um volume. O volume terá os aspetos definidos no pool escolhido.

Exemplo de classe de armazenamento

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=standard"
allowVolumeExpansion: true
```

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

- O primeiro StorageClass) (`cvs-extreme-extra-protection`mapeia para o primeiro pool virtual. Esse é o único pool que oferece desempenho extremo com uma reserva de snapshot de 10%.
- O último StorageClass) (`cvs-extra-protection`chama qualquer pool de armazenamento que forneça uma reserva de snapshot de 10%. O Astra Trident decide qual pool virtual está selecionado e garante que o requisito de reserva de snapshot seja atendido.

Exemplos de tipo de serviço CVS

Os exemplos a seguir fornecem exemplos de configurações para o tipo de serviço CVS.

Exemplo 1: Configuração mínima

Essa é a configuração mínima de back-end usada `storageClass` para especificar o tipo de serviço CVS e o nível de serviço padrão `standardsw`.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
```

```
  client_id: '12345678901234567890'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
serviceLevel: standardsw
```

Exemplo 2: Configuração do pool de armazenamento

Essa configuração de back-end de exemplo é usada `storagePools` para configurar um pool de armazenamento.

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDaT+Oui9FBAw19
    L1AGEkrYU5xd9K5NlO5jMkIFND5wCD+Nv+jd1GvtFRLaLK5RvXyF5wzvztmODNS+
    qtScpQ+5cFpQkuGtv9U9+N6qtuVYYO3b504Kp5CtqVPJCgMJaK2j8pZTIqUiMum/
    5/Y9oTbZrjAHSMgJm2nHzFq2X0rqVMaHghI6ATm4DOuWx8XGWKTGIPlc0qPqJlqS
    LLaWOH4VIZQZCAyW5IUp9CAmwqHgdG0uhFNfCgMmED6PBUvVLsLvcq86X+QSWR9k
    ETqElj/sGCenPF7ti1DhGBFafd9hPnxg9PZY29ArEZwY9G/ZjZQX7WPgs0VvxiNR
    DxZRC3GXAgMBAAECggEACn5c59bG/qnVEVI1CwMAalM5M2z09JFhlLlljKwntNPj
    Vilw2eTW2+UE7HbJru/S7KQgA5Dnn9kvCraEahPRuddUMrD0vG4kTl/IODV6uFuk
    Y0sZfbqd4jMUQ21smvGsqFzwloYWS5qzO1W83ivXH/HW/iqkmY2eW+EPRS/hwSSu
    SscR+SojI7PB0BWSJhlV4yqYf3vcD/D95el2CVHfRCkL85DKumeZ+yHEnpiXGZAE
    t8xSs4a5OOPm6NHhevCw2a/UQ95/foXNUR450HtbjieJo5o+FF6EYZQGfU2ZHZO8
    37FBKuaJkdGW5xqaI9TL7aqkGkFMF4F2qvOZM+vy8QKBgQD4oVuOkJDlhkTHP86W
    esFlw1kpWyJR9ZA7LI0g/rVpslnX+XdDq0WQf4umdLNau5hYEH9LU6ZSGs1Xk3/B
    NHwR6OXFuqEKNiu83d0zSlHhTy7PZpOZdj5a/vVvQfPDMz7OvsqLRd7YCAbdzuQ0
    +Ahq0Ztwvg0HQ64hdW0ukpYRRwKBgQDgyHj98oqswoYuIa+pP1yS0pPwLmjwKyNm
    /HayzCp+Qjiyy7Tzg8AUqlH1Ou83XbV428jvg7kDhO7PCCKFq+mMmfqHmTpb0Maq
    KpKnZg4ipsqP1yHNNEoRmcailXbwIhCLewMqMrggUiLOmCw4PscL5nK+4GKu2XE1
    jLqjWAZFMQKBgFHkQ9XXRAJ1kR3XpGHoGN890pZOkCVSrqju6aUef/5KYlFCt8ew
    F/+aIxM2iQSvmWQYOvVCnhuY/F2GFaQ7d0om3decuwI0CX/xy7PjHMkLXa2uaZs4
    WR17sLduj62RqXRLX0c0QkwBiNFyHbRcpdkZJQujbYMhBa+7j7SxT4BtAoGAWMWT
    UucocRXZm/pdvz9wteNH3YDWnJLMxm1KC06qMXbBoYrliY4sm3ywJWMC+iCd/H8A
    Gecxd/xVu5mA2L2N3KMq18Zhz8Th0G5DwKyDRJgOQ0Q46yuNXOoYEjlo4Wjyk8Me
    +tlQ8iK98E0UmZnhTgfSpSNElbz2AqnzQ3MN9uECgYAqdvdVPnKGfvdtZ2DjyMoJ
    E89UIC41WjjJGmHsd8W65+3X0RwMzKMT6aZc5tK9J5dHvmWIETnbM+lTImdBBFga
    NWOC6f3r2xbGXHhaWSl+nobpTuvlo56ZRJVvVk7lFMsiddzMuHH8pxfgNJemwA4P
    ThDHCejv035NNV6KyoO0tA==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
```

```
  client_id: '107071413297115343396'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

O que se segue?

Depois de criar o arquivo de configuração de back-end, execute o seguinte comando:

```
tridentctl create backend -f <backend-file>
```

Se a criação do backend falhar, algo está errado com a configuração do backend. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

Configurar um back-end NetApp HCI ou SolidFire

Saiba como criar e usar um back-end Element com sua instalação do Astra Trident.

Detalhes do driver do elemento

O Astra Trident fornece o `solidfire-san` driver de storage para se comunicar com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMuy* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

O `solidfire-san` driver de armazenamento suporta os modos de volume *file* e *block*. Para o `Filesystem` volumeMode, o Astra Trident cria um volume e cria um sistema de arquivos. O tipo de sistema de arquivos é especificado pelo StorageClass.

Condutor	Protocolo	Modo de volume	Modos de acesso suportados	Sistemas de arquivos suportados
`solidfire-san`	ISCSI	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de ficheiros. Dispositivo de bloco bruto.
`solidfire-san`	ISCSI	Sistema de ficheiros	RWO, RWOP	`xfs` `ext3`, , `ext4`

Antes de começar

Você precisará do seguinte antes de criar um backend de elemento.

- Um sistema de storage compatível que executa o software Element.
- Credenciais para um usuário de administrador ou locatário de cluster do NetApp HCI/SolidFire que possa gerenciar volumes.
- Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas iSCSI apropriadas instaladas. "informações sobre a preparação do nó de trabalho"Consulte a .

Opções de configuração de back-end

Consulte a tabela a seguir para obter as opções de configuração de back-end:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	Sempre "SolidFire-san"
`backendName`	Nome personalizado ou back-end de storage	Endereço IP "SolidFire_" e armazenamento (iSCSI)
`Endpoint`	MVIP para o cluster SolidFire com credenciais de locatário	
`SVIP`	Porta e endereço IP de armazenamento (iSCSI)	
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes.	""
`TenantName`	Nome do locatário a utilizar (criado se não for encontrado)	
`InitiatorIFace`	Restringir o tráfego iSCSI a uma interface de host específica	"padrão"
`UseCHAP`	Use CHAP para autenticar iSCSI. O Astra Trident usa CHAP.	verdadeiro
`AccessGroups`	Lista de IDs de Grupo de Acesso a utilizar	Encontra a ID de um grupo de acesso chamado "Trident"

Parâmetro	Descrição	Padrão
`Types`	Especificações de QoS	
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor	"" (não aplicado por padrão)
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Por exemplo, "api":false, "método":true"	nulo

Não use `debugTraceFlags` a menos que você esteja solucionando problemas e exija um despejo de log detalhado.

Exemplo 1: Configuração de back-end para `solidfire-san` driver com três tipos de volume

Este exemplo mostra um arquivo de back-end usando autenticação CHAP e modelagem de três tipos de volume com garantias de QoS específicas. Provavelmente você definiria classes de armazenamento para consumir cada uma delas usando o `IOPS` parâmetro de classe de armazenamento.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
```

Exemplo 2: Configuração de classe de back-end e armazenamento para `solidfire-san` driver com pools virtuais

Este exemplo mostra o arquivo de definição de back-end configurado com pools virtuais junto com o StorageClasses que se referem a eles.

O Astra Trident copia rótulos presentes em um pool de storage para a LUN de storage de back-end no provisionamento. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

No arquivo de definição de back-end de exemplo mostrado abaixo, padrões específicos são definidos para todos os pools de armazenamento, que definem o `type` em Prata. Os pools virtuais são definidos na `storage` seção. Neste exemplo, alguns dos pools de armazenamento definem seu próprio tipo, e alguns pools substituem os valores padrão definidos acima.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
- labels:
    performance: gold
    cost: '4'
  zone: us-east-1a
```

```
  type: Gold
- labels:
    performance: silver
    cost: '3'
  zone: us-east-1b
  type: Silver
- labels:
    performance: bronze
    cost: '2'
  zone: us-east-1c
  type: Bronze
- labels:
    performance: silver
    cost: '1'
  zone: us-east-1d
```

As seguintes definições do StorageClass referem-se aos pools virtuais acima. Usando o `parameters.selector` campo, cada StorageClass chama qual(s) pool(s) virtual(s) pode(m) ser(ão) usado(s) para hospedar um volume. O volume terá os aspetos definidos no pool virtual escolhido.

O primeiro StorageClass) (`solidfire-gold-four`será mapeado para o primeiro pool virtual. Este é o único pool que oferece desempenho de ouro com um `Volume Type QoS` de ouro. O último StorageClass) (`solidfire-silver`chama qualquer pool de armazenamento que ofereça um desempenho prateado. O Astra Trident decidirá qual pool virtual está selecionado e garantirá que o requisito de storage seja atendido.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold; cost=4"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=3"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze; cost=2"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=1"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
  fsType: "ext4"
```

Encontre mais informações

- "Grupos de acesso de volume"

Controladores SAN ONTAP

Descrição geral do controlador SAN ONTAP

Saiba mais sobre como configurar um back-end ONTAP com drivers SAN ONTAP e Cloud Volumes ONTAP.

Detalhes do driver SAN ONTAP

O Astra Trident fornece os seguintes drivers de storage SAN para se comunicar com o cluster ONTAP. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMuy* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Se você estiver usando o Astra Control para proteção, recuperação e mobilidade, leia Compatibilidade com driver Astra Control.

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-san`	ISCSI	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san`	ISCSI	Sistema de ficheiros	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume do sistema de arquivos.	`xfs` `ext3`, , `ext4`
`ontap-san`	NVMe/TCP Considerações adicionais para NVMe/TCP Consulte a .	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san`	NVMe/TCP Considerações adicionais para NVMe/TCP Consulte a .	Sistema de ficheiros	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume do sistema de arquivos.	`xfs` `ext3`, , `ext4`

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-san-economy`	ISCSI	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san-economy`	ISCSI	Sistema de ficheiros	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume do sistema de arquivos.	`xfs` `ext3`, , `ext4`

Compatibilidade com driver Astra Control

O Astra Control oferece proteção aprimorada, recuperação de desastres e mobilidade (migrando volumes entre clusters Kubernetes) para volumes criados com os `ontap-nas` drivers , `ontap-nas-flexgroup` e . `ontap-san` "Pré-requisitos de replicação do Astra Control"Consulte para obter detalhes.

- Use `ontap-san-economy` somente se a contagem de uso de volume persistente for esperada ser maior que "Limites de volume ONTAP suportados".
- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente for esperada para ser maior do que "Limites de volume ONTAP suportados" e o `ontap-san-economy` driver não puder ser usado.
- Não use o uso `ontap-nas-economy` se você antecipar a necessidade de proteção de dados, recuperação de desastres ou mobilidade.

Permissões do usuário

O Astra Trident espera ser executado como administrador da ONTAP ou SVM, normalmente usando o `admin` usuário do cluster ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. Para implantações do Amazon FSX for NetApp ONTAP, o Astra Trident espera ser executado como administrador do ONTAP ou SVM, usando o usuário do cluster `fsxadmin` ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. O `fsxadmin` usuário é um substituto limitado para o usuário administrador do cluster.

Se você usar o `limitAggregateUsage` parâmetro, as permissões de administrador do cluster serão necessárias. Ao usar o Amazon FSX for NetApp ONTAP com Astra Trident, o `limitAggregateUsage` parâmetro não funcionará com as `vsadmin` contas de usuário e `fsxadmin`. A operação de configuração falhará se você especificar este parâmetro.

Embora seja possível criar uma função mais restritiva no ONTAP que um driver Trident pode usar, não recomendamos. A maioria das novas versões do Trident chamarão APIs adicionais que teriam que ser contabilizadas, tornando as atualizações difíceis e suscetíveis a erros.

Considerações adicionais para NVMe/TCP

O Astra Trident dá suporte ao protocolo NVMe (non-volátil Memory Express) usando `ontap-san` o driver, incluindo:

- IPv6
- Snapshots e clones de volumes NVMe
- Redimensionamento de um volume NVMe
- Importação de um volume NVMe que foi criado fora do Astra Trident para que seu ciclo de vida possa ser gerenciado pelo Astra Trident
- Multipathing nativo NVMe
- Desligamento gracioso ou vergonhoso dos K8s nós (24,02)

O Astra Trident não é compatível com:

- DH-HMAC-CHAP que é suportado nativamente pelo NVMe
- Multipathing de mapeador de dispositivos (DM)
- Criptografia LUKS

Prepare-se para configurar o back-end com drivers SAN ONTAP

Entenda os requisitos e as opções de autenticação para configurar um back-end do ONTAP com drivers de SAN ONTAP.

Requisitos

Para todos os back-ends ONTAP, o Astra Trident requer pelo menos um agregado atribuído ao SVM.

Lembre-se de que você também pode executar mais de um driver e criar classes de armazenamento que apontam para um ou outro. Por exemplo, você pode configurar uma `san-dev` classe que usa o `ontap-san` driver e uma `san-default` classe que usa a `ontap-san-economy` mesma.

Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas iSCSI apropriadas instaladas. "Prepare o nó de trabalho"Consulte para obter detalhes.

Autenticar o back-end do ONTAP

O Astra Trident oferece dois modos de autenticação no back-end do ONTAP.

- Baseado em credenciais: O nome de usuário e senha para um usuário do ONTAP com as permissões necessárias. Recomenda-se a utilização de uma função de início de sessão de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.
- Baseado em certificado: O Astra Trident também pode se comunicar com um cluster ONTAP usando um certificado instalado no back-end. Aqui, a definição de back-end deve conter valores codificados em Base64 do certificado de cliente, chave e certificado de CA confiável, se usado (recomendado).

Você pode atualizar os backends existentes para mover entre métodos baseados em credenciais e baseados em certificado. No entanto, apenas um método de autenticação é suportado por vez. Para alternar para um método de autenticação diferente, você deve remover o método existente da configuração de back-end.

Se você tentar fornecer **credenciais e certificados**, a criação de back-end falhará com um erro que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ative a autenticação baseada em credenciais

O Astra Trident requer as credenciais para um administrador com escopo SVM/cluster para se comunicar com o back-end do ONTAP. Recomenda-se a utilização de funções padrão predefinidas, como `admin` ou `vsadmin`. Isso garante compatibilidade direta com futuras versões do ONTAP que podem expor APIs de recursos a serem usadas por futuras versões do Astra Trident. Uma função de login de segurança personalizada pode ser criada e usada com o Astra Trident, mas não é recomendada.

Uma definição de backend de exemplo será assim:

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

Tenha em mente que a definição de back-end é o único lugar onde as credenciais são armazenadas em texto simples. Depois que o back-end é criado, os nomes de usuário/senhas são codificados com Base64 e armazenados como segredos do Kubernetes. A criação ou atualização de um backend é a única etapa que requer conhecimento das credenciais. Como tal, é uma operação somente de administrador, a ser realizada pelo administrador do Kubernetes/storage.

Ativar autenticação baseada em certificado

Backends novos e existentes podem usar um certificado e se comunicar com o back-end do ONTAP. Três parâmetros são necessários na definição de backend.

- ClientCertificate: Valor codificado base64 do certificado do cliente.
- ClientPrivateKey: Valor codificado em base64 da chave privada associada.
- TrustedCACertificate: Valor codificado base64 do certificado CA confiável. Se estiver usando uma CA

confiável, esse parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as etapas a seguir.

Passos

1. Gerar um certificado e chave de cliente. Ao gerar, defina Nome Comum (CN) para o usuário ONTAP para autenticar como.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. Adicionar certificado de CA confiável ao cluster do ONTAP. Isso pode já ser Tratado pelo administrador do armazenamento. Ignore se nenhuma CA confiável for usada.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Instale o certificado e a chave do cliente (a partir do passo 1) no cluster do ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Confirme se a função de login de segurança do ONTAP suporta `cert` o método de autenticação.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. Teste a autenticação usando certificado gerado. Substitua o ONTAP Management LIF> e o <vserver name> por IP de LIF de gerenciamento e nome da SVM.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codificar certificado, chave e certificado CA confiável com Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Crie backend usando os valores obtidos na etapa anterior.

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Atualizar métodos de autenticação ou girar credenciais

Você pode atualizar um back-end existente para usar um método de autenticação diferente ou para girar suas credenciais. Isso funciona de ambas as maneiras: Backends que fazem uso de nome de usuário / senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para nome de usuário / senha com base. Para fazer isso, você deve remover o método de autenticação existente e adicionar o novo método de autenticação. Em seguida, use o arquivo backend.json atualizado contendo os parâmetros necessários para executar `tridentctl backend update`.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Ao girar senhas, o administrador de armazenamento deve primeiro atualizar a senha do usuário no ONTAP. Isso é seguido por uma atualização de back-end. Ao girar certificados, vários certificados podem ser adicionados ao usuário. O back-end é então atualizado para usar o novo certificado, seguindo o qual o certificado antigo pode ser excluído do cluster do ONTAP.

A atualização de um back-end não interrompe o acesso a volumes que já foram criados, nem afeta as conexões de volume feitas depois. Uma atualização de back-end bem-sucedida indica que o Astra Trident pode se comunicar com o back-end do ONTAP e lidar com operações de volume futuras.

Autentique conexões com CHAP bidirecional

O Astra Trident pode autenticar sessões iSCSI com CHAP bidirecional para os `ontap-san` drivers e `ontap-san-economy`. Isso requer a ativação da `useCHAP` opção na definição de backend. Quando definido como `true`, o Astra Trident configura a segurança do iniciador padrão do SVM para CHAP bidirecional e define o nome de usuário e os segredos do arquivo de back-end. O NetApp recomenda o uso de CHAP bidirecional para autenticar conexões. Veja a seguinte configuração de exemplo:

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

O `useCHAP` parâmetro é uma opção booleana que pode ser configurada apenas uma vez. Ele é definido como false por padrão. Depois de configurá-lo como verdadeiro, você não pode configurá-lo como falso.

Além `useCHAP=true` do , os `chapInitiatorSecret` campos , `chapTargetInitiatorSecret`, `chapTargetUsername`, e `chapUsername` devem ser incluídos na definição de back-end. Os segredos podem ser alterados depois que um backend é criado executando `tridentctl update`.

Como funciona

Ao definir `useCHAP` como verdadeiro, o administrador de storage instrui o Astra Trident a configurar o CHAP no back-end de storage. Isso inclui o seguinte:

- Configuração do CHAP no SVM:
  - Se o tipo de segurança do iniciador padrão da SVM for nenhum (definido por padrão) **e** não houver LUNs pré-existentes no volume, o Astra Trident definirá o tipo de segurança padrão `CHAP` e continuará configurando o iniciador CHAP e o nome de usuário e os segredos de destino.
  - Se o SVM contiver LUNs, o Astra Trident não ativará o CHAP no SVM. Isso garante que o acesso a LUNs que já estão presentes no SVM não seja restrito.
- Configurando o iniciador CHAP e o nome de usuário e os segredos de destino; essas opções devem ser especificadas na configuração de back-end (como mostrado acima).

Depois que o back-end é criado, o Astra Trident cria um CRD correspondente `tridentbackend` e armazena os segredos e nomes de usuário do CHAP como segredos do Kubernetes. Todos os PVS criados pelo Astra Trident neste back-end serão montados e anexados através do CHAP.

Gire credenciais e atualize os backends

Você pode atualizar as credenciais CHAP atualizando os parâmetros CHAP no `backend.json` arquivo. Isso exigirá a atualização dos segredos CHAP e o uso do `tridentctl update` comando para refletir essas alterações.

Ao atualizar os segredos CHAP para um backend, você deve usar `tridentctl` para atualizar o backend. Não atualize as credenciais no cluster de storage por meio da IU da CLI/ONTAP, pois o Astra Trident não conseguirá aceitar essas alterações.

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

As conexões existentes não serão afetadas. Elas continuarão ativas se as credenciais forem atualizadas pelo Astra Trident no SVM. As novas conexões usarão as credenciais atualizadas e as conexões existentes continuam ativas. Desconetar e reconetar PVS antigos resultará em eles usando as credenciais atualizadas.

Exemplos e opções de configuração de SAN ONTAP

Saiba como criar e usar drivers SAN ONTAP com sua instalação do Astra Trident. Esta seção fornece exemplos de configuração de back-end e detalhes para mapear backends para StorageClasses.

Opções de configuração de back-end

Consulte a tabela a seguir para obter as opções de configuração de back-end:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	`ontap-nas` `ontap-nas-economy`, , `ontap-nas-flexgroup` `ontap-san` , , , `ontap-san-economy`
`backendName`	Nome personalizado ou back-end de storage	Nome do driver e dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM. Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . Para o switchover MetroCluster otimizado, consulte o Exemplo de MetroCluster.	"10,0.0,1", "[2001:1234:abcd::fefe]"
`dataLIF`	Endereço IP do protocolo LIF. **Não especifique para iSCSI.** O Astra Trident usa "Mapa de LUN seletivo da ONTAP" para descobrir os LIFs iSCI necessários para estabelecer uma sessão de vários caminhos. Um aviso é gerado se `dataLIF` for definido explicitamente. **Omita para MetroCluster.** Consulte Exemplo de MetroCluster.	Derivado do SVM
`svm`	Máquina virtual de armazenamento para usar **omit for MetroCluster.** Consulte Exemplo de MetroCluster.	Derivado se uma SVM `managementLIF` for especificada
`useCHAP`	Use CHAP para autenticar iSCSI para drivers SAN ONTAP [Boolean]. Defina como `true` para o Astra Trident para configurar e usar CHAP bidirecional como a autenticação padrão para o SVM dado no back-end. "Prepare-se para configurar o back-end com drivers SAN ONTAP"Consulte para obter detalhes.	`false`
`chapInitiatorSecret`	Segredo do iniciador CHAP. Necessário se `useCHAP=true`	""
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""
`chapTargetInitiatorSecret`	Segredo do iniciador de destino CHAP. Necessário se `useCHAP=true`	""
`chapUsername`	Nome de utilizador de entrada. Necessário se `useCHAP=true`	""
`chapTargetUsername`	Nome de utilizador alvo. Necessário se `useCHAP=true`	""
`clientCertificate`	Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado	""

Parâmetro	Descrição	Padrão
`clientPrivateKey`	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
`username`	Nome de usuário necessário para se comunicar com o cluster ONTAP. Usado para autenticação baseada em credenciais.	""
`password`	Senha necessária para se comunicar com o cluster ONTAP. Usado para autenticação baseada em credenciais.	""
`svm`	Máquina virtual de armazenamento para usar	Derivado se uma SVM `managementLIF` for especificada
`storagePrefix`	Prefixo usado ao provisionar novos volumes na SVM. Não pode ser modificado mais tarde. Para atualizar esse parâmetro, você precisará criar um novo backend.	`trident`
`limitAggregateUsage`	Falha no provisionamento se o uso estiver acima dessa porcentagem. Se você estiver usando um back-end do Amazon FSX for NetApp ONTAP, não `limitAggregateUsage`especifique . O fornecido `fsxadmin` e `vsadmin` não contém as permissões necessárias para recuperar o uso agregado e limitá-lo usando o Astra Trident.	"" (não aplicado por padrão)
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor. Também restringe o tamanho máximo dos volumes que gerencia para qtrees e LUNs.	"" (não aplicado por padrão)
`lunsPerFlexvol`	Máximo de LUNs por FlexVol, tem de estar no intervalo [50, 200]	`100`
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Por exemplo, não use a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	`null`

Parâmetro	Descrição	Padrão
`useREST`	Parâmetro booleano para usar APIs REST do ONTAP. **A visualização técnica** `useREST` é fornecida como uma **prévia técnica** que é recomendada para ambientes de teste e não para cargas de trabalho de produção. Quando definido como `true`, o Astra Trident usará as APIS REST do ONTAP para se comunicar com o back-end. Esse recurso requer o ONTAP 9.11,1 e posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao `ontap` aplicativo. Isso é satisfeito com as funções e `cluster-admin` predefinidas `vsadmin`. `useREST` Não é suportado com MetroCluster. `useREST` É totalmente qualificado para NVMe/TCP.	`false`
`sanType`	Utilize para selecionar `iscsi` para iSCSI ou `nvme` para NVMe/TCP.	`iscsi` se estiver em branco

Opções de configuração de back-end para volumes de provisionamento

Você pode controlar o provisionamento padrão usando essas opções na `defaults` seção da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para LUNs	"verdadeiro"
`spaceReserve`	Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso)	"nenhum"
`snapshotPolicy`	Política de instantâneos a utilizar	"nenhum"
`qosPolicy`	Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend. O uso de grupos de política de QoS com o Astra Trident requer o ONTAP 9.8 ou posterior. Recomendamos o uso de um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de política de QoS compartilhado aplicará o limite máximo da taxa de transferência total de todos os workloads.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend	""
`snapshotReserve`	Porcentagem de volume reservado para snapshots	"0" se `snapshotPolicy` for "nenhum", caso contrário ""
`splitOnClone`	Divida um clone de seu pai na criação	"falso"

Parâmetro	Descrição	Padrão
`encryption`	Ative a criptografia de volume do NetApp (NVE) no novo volume; o padrão é `false`. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se o NAE estiver ativado no back-end, qualquer volume provisionado no Astra Trident será o NAE ativado. Para obter mais informações, consulte: "Como o Astra Trident funciona com NVE e NAE".	"falso"
`luksEncryption`	Ativar encriptação LUKS. "Usar a configuração de chave unificada do Linux (LUKS)"Consulte a . A criptografia LUKS não é compatível com NVMe/TCP.	""
`securityStyle`	Estilo de segurança para novos volumes	`unix`
`tieringPolicy`	Política de disposição em camadas para usar "nenhuma"	"Somente snapshot" para configuração pré-ONTAP 9.5 SVM-DR

Exemplos de provisionamento de volume

Aqui está um exemplo com padrões definidos:

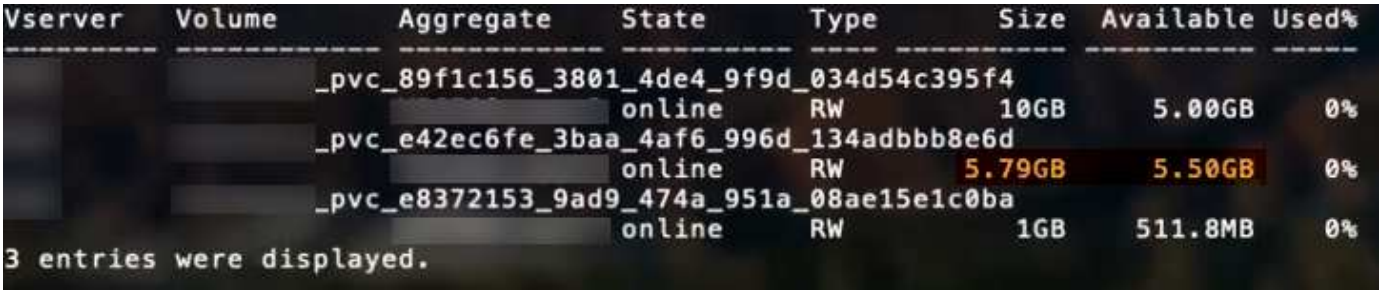
```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

Para todos os volumes criados com `ontap-san` o driver, o Astra Trident adiciona uma capacidade extra de 10% ao FlexVol para acomodar os metadados do LUN. O LUN será provisionado com o tamanho exato que o usuário solicita no PVC. O Astra Trident adiciona 10% ao FlexVol (mostra como tamanho disponível no ONTAP). Os usuários agora terão a capacidade utilizável que solicitaram. Essa alteração também impede que LUNs fiquem somente leitura, a menos que o espaço disponível seja totalmente utilizado. Isto não se aplica à ONTAP-san-economia.

Para backends que definem `snapshotReserve`o , o Astra Trident calcula o tamanho dos volumes da seguinte forma:

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

O 1,1 é o 10% adicional que o Astra Trident adiciona ao FlexVol para acomodar os metadados do LUN. Para `snapshotReserve` 5%, e o pedido de PVC é de 5GiB, o tamanho total do volume é de 5,79GiB e o tamanho disponível é de 5,5GiB. O `volume show` comando deve mostrar resultados semelhantes a este exemplo:



Atualmente, o redimensionamento é a única maneira de usar o novo cálculo para um volume existente.

Exemplos mínimos de configuração

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros padrão. Esta é a maneira mais fácil de definir um backend.

Se você estiver usando o Amazon FSX no NetApp ONTAP com Astra Trident, recomendamos que você especifique nomes DNS para LIFs em vez de endereços IP.

Exemplo de SAN ONTAP

Esta é uma configuração básica usando `ontap-san` o driver.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

Exemplo de economia de SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

Exemplo de MetroCluster

Você pode configurar o back-end para evitar ter que atualizar manualmente a definição do back-end após o switchover e o switchback durante "Replicação e recuperação da SVM"o .

Para comutação e switchback contínuos, especifique o SVM usando `managementLIF` e omite os `dataLIF` parâmetros e. `svm` Por exemplo:

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Exemplo de autenticação baseada em certificado

Neste exemplo de configuração básica `clientCertificate` , `clientPrivateKey` e `trustedCACertificate` (opcional, se estiver usando CA confiável) são preenchidos `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado de CA confiável, respetivamente.

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

Exemplos CHAP bidirecional

Esses exemplos criam um backend com `useCHAP` definido como `true`.

Exemplo de ONTAP SAN CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Exemplo de CHAP de economia de SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Exemplo de NVMe/TCP

Você precisa ter um SVM configurado com NVMe no back-end do ONTAP. Esta é uma configuração básica de back-end para NVMe/TCP.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

Exemplos de backends com pools virtuais

Nesses arquivos de definição de back-end de exemplo, padrões específicos são definidos para todos os pools de armazenamento, como `spaceReserve` em nenhum, `spaceAllocation` em falso e `encryption` em falso. Os pools virtuais são definidos na seção armazenamento.

O Astra Trident define rótulos de provisionamento no campo "Comentários". Os comentários são definidos no FlexVol. O Astra Trident copia todas as etiquetas presentes em um pool virtual para o volume de storage no provisionamento. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

Nesses exemplos, alguns dos pools de armazenamento definem seus próprios `spaceReserve` `spaceAllocation` valores , e `encryption` , e alguns pools substituem os valores padrão.

Exemplo de SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
- labels:
    protection: gold
    creditpoints: '40000'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
    adaptiveQosPolicy: adaptive-extreme
- labels:
    protection: silver
    creditpoints: '20000'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
    qosPolicy: premium
- labels:
    protection: bronze
    creditpoints: '5000'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

Exemplo de economia de SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
labels:
  store: san_economy_store
region: us_east_1
storage:
- labels:
    app: oracledb
    cost: '30'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
- labels:
    app: postgresdb
    cost: '20'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
- labels:
    app: mysqldb
    cost: '10'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
- labels:
    department: legal
    creditpoints: '5000'
  zone: us_east_1c
```

```
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

Exemplo de NVMe/TCP

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: 'false'
  encryption: 'true'
storage:
- labels:
    app: testApp
    cost: '20'
  defaults:
    spaceAllocation: 'false'
    encryption: 'false'
```

Mapeie os backends para StorageClasses

As seguintes definições do StorageClass referem-se ao Exemplos de backends com pools virtuais. Usando o `parameters.selector` campo, cada StorageClass chama quais pools virtuais podem ser usados para hospedar um volume. O volume terá os aspetos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass será mapeado para o primeiro pool virtual `ontap-san` no back-end. Esta é a única piscina que oferece proteção de nível dourado.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- O `protection-not-gold` StorageClass será mapeado para o segundo e terceiro pool virtual no `ontap-san` back-end. Estas são as únicas piscinas que oferecem um nível de proteção diferente do ouro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- O `app-mysqldb` StorageClass será mapeado para o terceiro pool virtual no `ontap-san-economy` back-end. Este é o único pool que oferece configuração de pool de armazenamento para o aplicativo tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- O `protection-silver-creditpoints-20k` StorageClass será mapeado para o segundo pool virtual no `ontap-san` back-end. Esta é a única piscina que oferece proteção de nível de prata e 20000 pontos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- O `creditpoints-5k` StorageClass será mapeado para o terceiro pool virtual no `ontap-san` back-end e o quarto pool virtual no `ontap-san-economy` back-end. Estas são as únicas ofertas de pool com 5000 pontos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- O `my-test-app-sc` StorageClass será mapeado para o `testAPP` pool virtual no `ontap-san` driver com `sanType: nvme`o . Esta é a única piscina que oferece `testApp`.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

O Astra Trident decidirá qual pool virtual está selecionado e garantirá que o requisito de storage seja atendido.

Drivers nas ONTAP

Descrição geral do controlador ONTAP nas

Saiba mais sobre como configurar um back-end ONTAP com drivers nas ONTAP e Cloud Volumes ONTAP.

Detalhes do driver nas do ONTAP

O Astra Trident fornece os seguintes drivers de storage nas para se comunicar com o cluster ONTAP. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMuy* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Se você estiver usando o Astra Control para proteção, recuperação e mobilidade, leia Compatibilidade com driver Astra Control.

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-nas`	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	"" `nfs`, , `smb`

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-nas-economy`	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	"" `nfs`, , `smb`
`ontap-nas-flexgroup`	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	"" `nfs`, , `smb`

Compatibilidade com driver Astra Control

O Astra Control oferece proteção aprimorada, recuperação de desastres e mobilidade (migrando volumes entre clusters Kubernetes) para volumes criados com os `ontap-nas` drivers , `ontap-nas-flexgroup` e . `ontap-san` "Pré-requisitos de replicação do Astra Control"Consulte para obter detalhes.

- Use `ontap-san-economy` somente se a contagem de uso de volume persistente for esperada ser maior que "Limites de volume ONTAP suportados".
- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente for esperada para ser maior do que "Limites de volume ONTAP suportados" e o `ontap-san-economy` driver não puder ser usado.
- Não use o uso `ontap-nas-economy` se você antecipar a necessidade de proteção de dados, recuperação de desastres ou mobilidade.

Permissões do usuário

O Astra Trident espera ser executado como administrador da ONTAP ou SVM, normalmente usando o `admin` usuário do cluster ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função.

Para implantações do Amazon FSX for NetApp ONTAP, o Astra Trident espera ser executado como administrador do ONTAP ou SVM, usando o usuário do cluster `fsxadmin` ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. O `fsxadmin` usuário é um substituto limitado para o usuário administrador do cluster.

Se você usar o `limitAggregateUsage` parâmetro, as permissões de administrador do cluster serão necessárias. Ao usar o Amazon FSX for NetApp ONTAP com Astra Trident, o `limitAggregateUsage` parâmetro não funcionará com as `vsadmin` contas de usuário e `fsxadmin`. A operação de configuração falhará se você especificar este parâmetro.

Embora seja possível criar uma função mais restritiva no ONTAP que um driver Trident pode usar, não recomendamos. A maioria das novas versões do Trident chamarão APIs adicionais que teriam que ser contabilizadas, tornando as atualizações difíceis e suscetíveis a erros.

Prepare-se para configurar um back-end com drivers nas ONTAP

Entenda os requisitos, as opções de autenticação e as políticas de exportação para configurar um back-end do ONTAP com drivers nas do ONTAP.

Requisitos

- Para todos os back-ends ONTAP, o Astra Trident requer pelo menos um agregado atribuído ao SVM.
- Você pode executar mais de um driver e criar classes de armazenamento que apontam para um ou outro. Por exemplo, você pode configurar uma classe Gold que usa o `ontap-nas` driver e uma classe Bronze que usa o `ontap-nas-economy` um.
- Todos os seus nós de trabalho do Kubernetes precisam ter as ferramentas NFS apropriadas instaladas. "aqui"Consulte para obter mais detalhes.
- O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows. Prepare-se para provisionar volumes SMBConsulte para obter detalhes.

Autenticar o back-end do ONTAP

O Astra Trident oferece dois modos de autenticação no back-end do ONTAP.

- Baseado em credenciais: Esse modo requer permissões suficientes para o back-end do ONTAP. Recomenda-se usar uma conta associada a uma função de login de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.
- Baseado em certificado: Esse modo requer que um certificado seja instalado no back-end para que o Astra Trident se comunique com um cluster ONTAP. Aqui, a definição de back-end deve conter valores codificados em Base64 do certificado de cliente, chave e certificado de CA confiável, se usado (recomendado).

Você pode atualizar os backends existentes para mover entre métodos baseados em credenciais e baseados em certificado. No entanto, apenas um método de autenticação é suportado por vez. Para alternar para um método de autenticação diferente, você deve remover o método existente da configuração de back-end.

> Se você tentar fornecer **credenciais e certificados**, a criação de back-end falhará com um erro que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ative a autenticação baseada em credenciais

O Astra Trident requer as credenciais para um administrador com escopo SVM/cluster para se comunicar com o back-end do ONTAP. Recomenda-se a utilização de funções padrão predefinidas, como `admin` ou `vsadmin`. Isso garante compatibilidade direta com futuras versões do ONTAP que podem expor APIs de recursos a serem usadas por futuras versões do Astra Trident. Uma função de login de segurança personalizada pode ser criada e usada com o Astra Trident, mas não é recomendada.

Uma definição de backend de exemplo será assim:

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

Tenha em mente que a definição de back-end é o único lugar onde as credenciais são armazenadas em texto simples. Depois que o back-end é criado, os nomes de usuário/senhas são codificados com Base64 e armazenados como segredos do Kubernetes. A criação/updation de um backend é a única etapa que requer conhecimento das credenciais. Como tal, é uma operação somente de administrador, a ser realizada pelo administrador do Kubernetes/storage.

Ativar autenticação baseada em certificado

Backends novos e existentes podem usar um certificado e se comunicar com o back-end do ONTAP. Três parâmetros são necessários na definição de backend.

- ClientCertificate: Valor codificado base64 do certificado do cliente.
- ClientPrivateKey: Valor codificado em base64 da chave privada associada.
- TrustedCACertificate: Valor codificado base64 do certificado CA confiável. Se estiver usando uma CA confiável, esse parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as etapas a seguir.

Passos

1. Gerar um certificado e chave de cliente. Ao gerar, defina Nome Comum (CN) para o usuário ONTAP para autenticar como.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. Adicionar certificado de CA confiável ao cluster do ONTAP. Isso pode já ser Tratado pelo administrador do armazenamento. Ignore se nenhuma CA confiável for usada.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Instale o certificado e a chave do cliente (a partir do passo 1) no cluster do ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Confirme se a função de login de segurança do ONTAP suporta `cert` o método de autenticação.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. Teste a autenticação usando certificado gerado. Substitua o ONTAP Management LIF> e o <vserver name> por IP de LIF de gerenciamento e nome da SVM. Você deve garantir que o LIF tenha sua política de serviço definida como `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codificar certificado, chave e certificado CA confiável com Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Crie backend usando os valores obtidos na etapa anterior.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Atualizar métodos de autenticação ou girar credenciais

Você pode atualizar um back-end existente para usar um método de autenticação diferente ou para girar suas credenciais. Isso funciona de ambas as maneiras: Backends que fazem uso de nome de usuário / senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para nome de usuário / senha com base. Para fazer isso, você deve remover o método de autenticação existente e adicionar o novo método de autenticação. Em seguida, use o arquivo backend.json atualizado contendo os parâmetros necessários para executar `tridentctl update backend`.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Ao girar senhas, o administrador de armazenamento deve primeiro atualizar a senha do usuário no ONTAP. Isso é seguido por uma atualização de back-end. Ao girar certificados, vários certificados podem ser adicionados ao usuário. O back-end é então atualizado para usar o novo certificado, seguindo o qual o certificado antigo pode ser excluído do cluster do ONTAP.

A atualização de um back-end não interrompe o acesso a volumes que já foram criados, nem afeta as conexões de volume feitas depois. Uma atualização de back-end bem-sucedida indica que o Astra Trident pode se comunicar com o back-end do ONTAP e lidar com operações de volume futuras.

Gerenciar políticas de exportação de NFS

O Astra Trident usa políticas de exportação de NFS para controlar o acesso aos volumes provisionados.

O Astra Trident oferece duas opções ao trabalhar com políticas de exportação:

- O Astra Trident pode gerenciar dinamicamente a própria política de exportação; nesse modo de operação, o administrador de armazenamento especifica uma lista de blocos CIDR que representam endereços IP admissíveis. O Astra Trident adiciona IPs de nós que se enquadram nesses intervalos à política de exportação automaticamente. Como alternativa, quando nenhum CIDR é especificado, qualquer IP unicast de escopo global encontrado nos nós será adicionado à política de exportação.

- Os administradores de storage podem criar uma política de exportação e adicionar regras manualmente. O Astra Trident usa a política de exportação padrão, a menos que um nome de política de exportação diferente seja especificado na configuração.

Gerencie dinamicamente políticas de exportação

O Astra Trident permite gerenciar dinamicamente políticas de exportação para back-ends ONTAP. Isso fornece ao administrador de armazenamento a capacidade de especificar um espaço de endereço permitido para IPs de nó de trabalho, em vez de definir regras explícitas manualmente. Ele simplifica muito o gerenciamento de políticas de exportação. As modificações na política de exportação não exigem mais intervenção manual no cluster de storage. Além disso, isso ajuda a restringir o acesso ao cluster de armazenamento somente aos nós de trabalho que têm IPs no intervalo especificado, suportando um gerenciamento refinado e automatizado.

Não use NAT (Network Address Translation) ao usar políticas de exportação dinâmicas. Com o NAT, o controlador de armazenamento vê o endereço NAT frontend e não o endereço IP real do host, portanto, o acesso será negado quando nenhuma correspondência for encontrada nas regras de exportação.

Exemplo

Há duas opções de configuração que devem ser usadas. Aqui está um exemplo de definição de backend:

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
- 192.168.0.0/24
autoExportPolicy: true
```

Ao usar esse recurso, você deve garantir que a junção raiz do SVM tenha uma política de exportação criada anteriormente com uma regra de exportação que permita o bloco CIDR do nó (como a política de exportação padrão). Siga sempre as práticas recomendadas recomendadas pela NetApp para dedicar um SVM para Astra Trident.

Aqui está uma explicação de como esse recurso funciona usando o exemplo acima:

- `autoExportPolicy` está definido como `true`. Isso indica que o Astra Trident criará uma política de exportação para `svm1` o SVM e tratará da adição e exclusão de regras usando `autoExportCIDRs` blocos de endereço. Por exemplo, um back-end com UUID 403b5326-8482-40dB-96d0-d83fb3f4daec e `autoExportPolicy` definido como `true` cria uma política de exportação nomeada `trident-403b5326-8482-40db-96d0-d83fb3f4daec` no SVM.
- `autoExportCIDRs` contém uma lista de blocos de endereços. Este campo é opcional e o padrão é ["0,0.0,0/0", "::/0"]. Se não estiver definido, o Astra Trident adiciona todos os endereços unicast de escopo

global encontrados nos nós de trabalho.

Neste exemplo, o `192.168.0.0/24` espaço de endereço é fornecido. Isso indica que os IPs de nós do Kubernetes que se enquadram nesse intervalo de endereços serão adicionados à política de exportação criada pelo Astra Trident. Quando o Astra Trident Registra um nó em que ele é executado, ele recupera os endereços IP do nó e os verifica em relação aos blocos de endereço fornecidos no `autoExportCIDRs`. depois de filtrar os IPs, o Astra Trident cria regras de política de exportação para os IPs de cliente que ele descobre, com uma regra para cada nó que identifica.

Você pode atualizar `autoExportPolicy` e `autoExportCIDRs` para backends depois de criá-los. Você pode anexar novos CIDR para um back-end que é gerenciado automaticamente ou excluir CIDR existentes. Tenha cuidado ao excluir CIDR para garantir que as conexões existentes não sejam descartadas. Você também pode optar por desativar `autoExportPolicy` um back-end e retornar a uma política de exportação criada manualmente. Isso exigirá a configuração do `exportPolicy` parâmetro em sua configuração de backend.

Depois que o Astra Trident criar ou atualizar um back-end, você pode verificar o back-end usando `tridentctl` ou o CRD correspondente `tridentbackend`:

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

Conforme os nós são adicionados a um cluster do Kubernetes e registrados na controladora Astra Trident, as políticas de exportação dos back-ends existentes são atualizadas (desde que elas estejam no intervalo de endereços especificado `autoExportCIDRs` no back-end).

Quando um nó é removido, o Astra Trident verifica todos os back-ends on-line para remover a regra de acesso do nó. Ao remover esse IP de nó das políticas de exportação de backends gerenciados, o Astra Trident impede montagens fraudulentas, a menos que esse IP seja reutilizado por um novo nó no cluster.

Para backends existentes anteriormente, a atualização do back-end com `tridentctl update backend` garantirá que o Astra Trident gerencie as políticas de exportação automaticamente. Isso criará uma nova

política de exportação nomeada após o UUID do backend e os volumes que estão presentes no backend usarão a política de exportação recém-criada quando forem montados novamente.

> A exclusão de um back-end com políticas de exportação gerenciadas automaticamente excluirá a política de exportação criada dinamicamente. Se o backend for recriado, ele será Tratado como um novo backend e resultará na criação de uma nova política de exportação.

Se o endereço IP de um nó ativo for atualizado, será necessário reiniciar o pod Astra Trident no nó. Em seguida, o Astra Trident atualizará a política de exportação para backends que ele conseguir refletir essa alteração de IP.

Prepare-se para provisionar volumes SMB

Com um pouco de preparação adicional, você pode provisionar volumes SMB usando `ontap-nas` drivers.

> Você precisa configurar os protocolos NFS e SMB/CIFS na SVM para criar um `ontap-nas-economy` volume SMB para ONTAP no local. A falha na configuração desses protocolos fará com que a criação de volume SMB falhe.

Antes de começar

Antes de provisionar volumes SMB, você deve ter o seguinte:

- Um cluster do Kubernetes com um nó de controlador Linux e pelo menos um nó de trabalho do Windows que executa o Windows Server 2019. O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows.
- Pelo menos um segredo do Astra Trident que contém suas credenciais do ative Directory. Para gerar segredo `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Um proxy CSI configurado como um serviço Windows. Para configurar um `csi-proxy`, "GitHub: CSI Proxy"consulte ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes executados no Windows.

Passos

1. Para o ONTAP no local, você pode criar, opcionalmente, um compartilhamento SMB ou o Astra Trident pode criar um para você.

   > Compartilhamentos SMB são necessários para o Amazon FSX for ONTAP.

   Você pode criar os compartilhamentos de administração SMB de duas maneiras usando o "Microsoft Management Console"snap-in pastas compartilhadas ou usando a CLI do ONTAP. Para criar compartilhamentos SMB usando a CLI do ONTAP:

   a. Se necessário, crie a estrutura do caminho do diretório para o compartilhamento.

      O `vserver cifs share create` comando verifica o caminho especificado na opção -path durante a criação de compartilhamento. Se o caminho especificado não existir, o comando falhará.

   b. Crie um compartilhamento SMB associado ao SVM especificado:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. Verifique se o compartilhamento foi criado:

```
vserver cifs share show -share-name share_name
```

"Crie um compartilhamento SMB"Consulte para obter detalhes completos.

2. Ao criar o back-end, você deve configurar o seguinte para especificar volumes SMB. Para obter todas as opções de configuração de back-end do FSX for ONTAP, "Opções e exemplos de configuração do FSX for ONTAP"consulte .

Parâmetro	Descrição	Exemplo
`smbShare`	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP; um nome para permitir que o Astra Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum ao compartilhamento aos volumes. Esse parâmetro é opcional para o ONTAP no local. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP e não pode ficar em branco.	`smb-share`
`nasType`	**Tem de estar definido para `smb`.** Se nulo, o padrão é `nfs`.	`smb`
`securityStyle`	Estilo de segurança para novos volumes. **Deve ser definido como `ntfs` ou `mixed` para volumes SMB.**	`ntfs` Ou `mixed` para volumes SMB
`unixPermissions`	Modo para novos volumes. **Deve ser deixado vazio para volumes SMB.**	""

Exemplos e opções de configuração do ONTAP nas

Aprenda a criar e usar drivers nas ONTAP com sua instalação do Astra Trident. Esta seção fornece exemplos de configuração de back-end e detalhes para mapear backends para StorageClasses.

Opções de configuração de back-end

Consulte a tabela a seguir para obter as opções de configuração de back-end:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	"ONTAP-nas", "ONTAP-nas-economy", "ONTAP-nas-FlexGroup", "ONTAP-san", "ONTAP-san-economy"
`backendName`	Nome personalizado ou back-end de storage	Nome do driver e dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . Para o switchover MetroCluster otimizado, consulte o Exemplo de MetroCluster.	"10,0.0,1", "[2001:1234:abcd::fefe]"
`dataLIF`	Endereço IP do protocolo LIF. Recomendamos especificar `dataLIF`. Se não for fornecido, o Astra Trident obtém LIFs de dados do SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) a ser usado para as operações de montagem NFS, permitindo que você crie um DNS de round-robin para balanceamento de carga em vários LIFs de dados. Pode ser alterado após a definição inicial. Consulte a . Pode ser definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . **Omita para MetroCluster.** Consulte Exemplo de MetroCluster.	Endereço especificado ou derivado do SVM, se não for especificado (não recomendado)
`svm`	Máquina virtual de armazenamento para usar **omit for MetroCluster.** Consulte Exemplo de MetroCluster.	Derivado se uma SVM `managementLIF` for especificada
`autoExportPolicy`	Ativar a criação e atualização automática da política de exportação [Boolean]. Com `autoExportPolicy` as opções e `autoExportCIDRs`, o Astra Trident pode gerenciar políticas de exportação automaticamente.	falso
`autoExportCIDRs`	Lista de CIDR para filtrar IPs de nós do Kubernetes quando `autoExportPolicy` está ativado. Com `autoExportPolicy` as opções e `autoExportCIDRs`, o Astra Trident pode gerenciar políticas de exportação automaticamente.	["0,0.0,0/0", "::/0"]»
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""
`clientCertificate`	Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado	""

Parâmetro	Descrição	Padrão
`clientPrivateKey`	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado	""
`username`	Nome de usuário para se conetar ao cluster/SVM. Usado para autenticação baseada em credenciais	
`password`	Senha para se conectar ao cluster/SVM. Usado para autenticação baseada em credenciais	
`storagePrefix`	Prefixo usado ao provisionar novos volumes na SVM. Não pode ser atualizado depois de configurá-lo	"Trident"
`limitAggregateUsage`	Falha no provisionamento se o uso estiver acima dessa porcentagem. **Não se aplica ao Amazon FSX for ONTAP**	"" (não aplicado por padrão)
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor. Também restringe o tamanho máximo dos volumes que gerencia para qtrees e LUNs, e a `qtreesPerFlexvol` opção permite personalizar o número máximo de qtrees por FlexVol.	"" (não aplicado por padrão)
`lunsPerFlexvol`	Máximo de LUNs por FlexVol, tem de estar no intervalo [50, 200]	"100"
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Por exemplo, não use `debugTraceFlags` a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. A configuração como null padrão para volumes NFS.	`nfs`
`nfsMountOptions`	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes normalmente são especificadas em classes de storage, mas se nenhuma opção de montagem for especificada em uma classe de storage, o Astra Trident voltará a usar as opções de montagem especificadas no arquivo de configuração do back-end de storage. Se nenhuma opção de montagem for especificada na classe de storage ou no arquivo de configuração, o Astra Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""
`qtreesPerFlexvol`	Qtrees máximos por FlexVol, têm de estar no intervalo [50, 300]	"200"

Parâmetro	Descrição	Padrão
`smbShare`	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP; um nome para permitir que o Astra Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum ao compartilhamento aos volumes. Esse parâmetro é opcional para o ONTAP no local. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP e não pode ficar em branco.	`smb-share`
`useREST`	Parâmetro booleano para usar APIs REST do ONTAP. **A visualização técnica** `useREST` é fornecida como uma **prévia técnica** que é recomendada para ambientes de teste e não para cargas de trabalho de produção. Quando definido como `true`, o Astra Trident usará as APIS REST do ONTAP para se comunicar com o back-end. Esse recurso requer o ONTAP 9.11,1 e posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao `ontap` aplicativo. Isso é satisfeito com as funções e `cluster-admin` predefinidas `vsadmin`. `useREST` Não é suportado com MetroCluster.	falso

Opções de configuração de back-end para volumes de provisionamento

Você pode controlar o provisionamento padrão usando essas opções na `defaults` seção da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para LUNs	"verdadeiro"
`spaceReserve`	Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso)	"nenhum"
`snapshotPolicy`	Política de instantâneos a utilizar	"nenhum"
`qosPolicy`	Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend. Não suportado pela ONTAP-nas-Economy.	""
`snapshotReserve`	Porcentagem de volume reservado para snapshots	"0" se `snapshotPolicy` for "nenhum", caso contrário ""
`splitOnClone`	Divida um clone de seu pai na criação	"falso"

Parâmetro	Descrição	Padrão
`encryption`	Ative a criptografia de volume do NetApp (NVE) no novo volume; o padrão é `false`. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se o NAE estiver ativado no back-end, qualquer volume provisionado no Astra Trident será o NAE ativado. Para obter mais informações, consulte: "Como o Astra Trident funciona com NVE e NAE".	"falso"
`tieringPolicy`	Política de disposição em camadas para usar "nenhuma"	"Somente snapshot" para configuração pré-ONTAP 9.5 SVM-DR
`unixPermissions`	Modo para novos volumes	"777" para volumes NFS; vazio (não aplicável) para volumes SMB
`snapshotDir`	Controla o acesso ao `.snapshot` diretório	"falso"
`exportPolicy`	Política de exportação a utilizar	"predefinição"
`securityStyle`	Estilo de segurança para novos volumes. Estilos de segurança e `unix` suporte de NFS `mixed`. Suporta SMB `mixed` e `ntfs` estilos de segurança.	O padrão NFS é `unix`. O padrão SMB é `ntfs`.

O uso de grupos de política de QoS com o Astra Trident requer o ONTAP 9.8 ou posterior. Recomenda-se usar um grupo de políticas QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de política de QoS compartilhado aplicará o limite máximo da taxa de transferência total de todos os workloads.

Exemplos de provisionamento de volume

Aqui está um exemplo com padrões definidos:

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: '10'
```

Para `ontap-nas` e `ontap-nas-flexgroups`, o Astra Trident agora usa um novo cálculo para garantir que o FlexVol seja dimensionado corretamente com a porcentagem de snapshotServe e PVC. Quando o usuário solicita um PVC, o Astra Trident cria o FlexVol original com mais espaço usando o novo cálculo. Esse cálculo garante que o usuário receba o espaço gravável que solicitou no PVC, e não menor espaço do que o que solicitou. Antes de v21,07, quando o usuário solicita um PVC (por exemplo, 5GiB), com o snapshotServe a 50 por cento, eles recebem apenas 2,5GiBMB de espaço gravável. Isso ocorre porque o que o usuário solicitou é todo o volume e `snapshotReserve` é uma porcentagem disso. Com o Trident 21,07, o que o usuário solicita é o espaço gravável e o Astra Trident define o `snapshotReserve` número como a porcentagem de todo o volume. Isto não se aplica `ontap-nas-economy` ao . Veja o exemplo a seguir para ver como isso funciona:

O cálculo é o seguinte:

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

Para snapshotServe de 50%, e a solicitação de PVC de 5GiB, o volume total é de 2/.5 10GiB e o tamanho disponível é de 5GiB, o que o usuário solicitou na solicitação de PVC. O `volume show` comando deve mostrar resultados semelhantes a este exemplo:

Os back-ends existentes de instalações anteriores provisionarão volumes conforme explicado acima ao atualizar o Astra Trident. Para volumes que você criou antes da atualização, você deve redimensionar seus volumes para que a alteração seja observada. Por exemplo, um PVC de 2GiB mm com `snapshotReserve=50` anterior resultou em um volume que fornece 1GiB GB de espaço gravável. Redimensionar o volume para 3GiB, por exemplo, fornece ao aplicativo 3GiBMB de espaço gravável em um volume de 6 GiB.

Exemplos mínimos de configuração

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros padrão. Esta é a maneira mais fácil de definir um backend.

> Se você estiver usando o Amazon FSX no NetApp ONTAP com Trident, a recomendação é especificar nomes DNS para LIFs em vez de endereços IP.

Exemplo de economia nas do ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemplo de ONTAP nas FlexGroup

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemplo de MetroCluster

Você pode configurar o back-end para evitar ter que atualizar manualmente a definição do back-end após o switchover e o switchback durante "Replicação e recuperação da SVM"o .

Para comutação e switchback contínuos, especifique o SVM usando `managementLIF` e omite os `dataLIF` parâmetros e. `svm` Por exemplo:

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Exemplo de volumes SMB

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemplo de autenticação baseada em certificado

Este é um exemplo de configuração de back-end mínimo. `clientCertificate`, `clientPrivateKey` E `trustedCACertificate` (opcional, se estiver usando CA confiável) são preenchidos `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado de CA confiável, respetivamente.

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Exemplo de política de exportação automática

Este exemplo mostra como você pode instruir o Astra Trident a usar políticas de exportação dinâmicas para criar e gerenciar a política de exportação automaticamente. Isso funciona da mesma forma para os `ontap-nas-economy` drivers e `ontap-nas-flexgroup`.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

Exemplo de endereços IPv6

Este exemplo mostra `managementLIF` usando um endereço IPv6.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

Exemplo do Amazon FSX para ONTAP usando volumes SMB

O `smbShare` parâmetro é necessário para o FSX for ONTAP usando volumes SMB.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Exemplos de backends com pools virtuais

Nos arquivos de definição de back-end de exemplo mostrados abaixo, padrões específicos são definidos para todos os pools de armazenamento, como `spaceReserve` em nenhum, `spaceAllocation` em falso e `encryption` em falso. Os pools virtuais são definidos na seção armazenamento.

O Astra Trident define rótulos de provisionamento no campo "Comentários". Os comentários são definidos no FlexVol for `ontap-nas` ou no FlexGroup `ontap-nas-flexgroup` for . O Astra Trident copia todas as etiquetas presentes em um pool virtual para o volume de storage no provisionamento. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

Nesses exemplos, alguns dos pools de armazenamento definem seus próprios `spaceReserve` `spaceAllocation` valores , e `encryption` , e alguns pools substituem os valores padrão.

Exemplo de ONTAP nas

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: 'false'
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
    app: msoffice
    cost: '100'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
    adaptiveQosPolicy: adaptive-premium
- labels:
    app: slack
    cost: '75'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    department: legal
    creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    app: wordpress
```

```
    cost: '50'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
    app: mysqldb
    cost: '25'
  zone: us_east_1d
  defaults:
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

Exemplo de ONTAP nas FlexGroup

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
    protection: gold
    creditpoints: '50000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    protection: gold
    creditpoints: '30000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    protection: silver
    creditpoints: '20000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
    protection: bronze
    creditpoints: '10000'
  zone: us_east_1d
  defaults:
```

```
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

Exemplo de economia nas do ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: nas_economy_store
region: us_east_1
storage:
- labels:
    department: finance
    creditpoints: '6000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    protection: bronze
    creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    department: engineering
    creditpoints: '3000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
    department: humanresource
    creditpoints: '2000'
  zone: us_east_1d
  defaults:
    spaceReserve: volume
```

```
    encryption: 'false'
    unixPermissions: '0775'
```

Mapeie os backends para StorageClasses

As seguintes definições do StorageClass referem-se Exemplos de backends com pools virtuaisa . Usando o `parameters.selector` campo, cada StorageClass chama quais pools virtuais podem ser usados para hospedar um volume. O volume terá os aspetos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass será mapeado para o primeiro e segundo pool virtual `ontap-nas-flexgroup` no back-end. Estas são as únicas piscinas que oferecem proteção de nível de ouro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- O `protection-not-gold` StorageClass será mapeado para o terceiro e quarto pool virtual no `ontap-nas-flexgroup` back-end. Estas são as únicas piscinas que oferecem um nível de proteção diferente do ouro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- O `app-mysqldb` StorageClass será mapeado para o quarto pool virtual `ontap-nas` no back-end. Este é o único pool que oferece configuração de pool de armazenamento para o aplicativo tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- O `protection-silver-creditpoints-20k` StorageClass será mapeado para o terceiro pool virtual no `ontap-nas-flexgroup` back-end. Esta é a única piscina que oferece proteção de nível de prata e 20000 pontos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- O `creditpoints-5k` StorageClass será mapeado para o terceiro pool virtual `ontap-nas` no back-end e o segundo pool virtual `ontap-nas-economy` no back-end. Estas são as únicas ofertas de pool com 5000 pontos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

O Astra Trident decidirá qual pool virtual está selecionado e garantirá que o requisito de storage seja atendido.

Atualização `dataLIF` após a configuração inicial

Você pode alterar o LIF de dados após a configuração inicial executando o seguinte comando para fornecer o novo arquivo JSON de back-end com LIF de dados atualizado.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

Se os PVCs estiverem anexados a um ou vários pods, você deverá reduzir todos os pods correspondentes e restaurá-los para que o novo LIF de dados entre em vigor.

Amazon FSX para NetApp ONTAP

Use o Astra Trident com o Amazon FSX para NetApp ONTAP

"Amazon FSX para NetApp ONTAP" É um serviço AWS totalmente gerenciado que permite que os clientes iniciem e executem sistemas de arquivos equipados com o sistema operacional de storage NetApp ONTAP. O FSX para ONTAP permite que você aproveite os recursos, o desempenho e os recursos administrativos do NetApp com os quais você já conhece, ao mesmo tempo em que aproveita a simplicidade, a agilidade, a segurança e a escalabilidade do armazenamento de dados na AWS. O FSX para ONTAP oferece suporte aos recursos do sistema de arquivos ONTAP e APIs de administração.

Visão geral

Um sistema de arquivos é o principal recurso do Amazon FSX, análogo a um cluster do ONTAP no local. Em cada SVM, você pode criar um ou vários volumes, que são contentores de dados que armazenam os arquivos e pastas em seu sistema de arquivos. Com o Amazon FSX for NetApp ONTAP, o Data ONTAP será fornecido como um sistema de arquivos gerenciado na nuvem. O novo tipo de sistema de arquivos é chamado de **NetApp ONTAP**.

Usando o Astra Trident com o Amazon FSX for NetApp ONTAP, você pode garantir que os clusters do Kubernetes executados no Amazon Elastic Kubernetes Service (EKS) provisionem volumes persistentes de bloco e arquivo com o respaldo do do ONTAP.

Considerações

- Volumes SMB:
  - Os volumes SMB são suportados usando `ontap-nas` apenas o driver.
  - Os volumes SMB não são compatíveis com o complemento Astra Trident EKS.
  - O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows.
- Antes do Astra Trident 24,02, os volumes criados nos sistemas de arquivos do Amazon FSX que têm backups automáticos ativados, não puderam ser excluídos pelo Trident. Para evitar esse problema no Astra Trident 24,02 ou posterior, especifique o `fsxFilesystemID`, `apiRegion` AWS , AWS `apikey` e AWS `secretKey` no arquivo de configuração de back-end do AWS FSX for ONTAP.

Se você estiver especificando uma função do IAM para o Astra Trident, poderá omitir especificar explicitamente os `apiRegion` campos , `apiKey` e `secretKey` para o Astra Trident. Para obter mais informações, "Opções e exemplos de configuração do FSX for ONTAP"consulte .

Detalhes do driver FSX for ONTAP

Você pode integrar o Astra Trident ao Amazon FSX for NetApp ONTAP usando os seguintes drivers:

- `ontap-san`: Cada PV provisionado é um LUN dentro de seu próprio volume do Amazon FSX for NetApp ONTAP.
- `ontap-san-economy`: Cada PV provisionado é um LUN com um número configurável de LUNs por volume do Amazon FSX for NetApp ONTAP.
- `ontap-nas`: Cada PV provisionado é um volume completo do Amazon FSX for NetApp ONTAP.
- `ontap-nas-economy`: Cada PV provisionado é uma qtree, com um número configurável de qtrees por volume do Amazon FSX for NetApp ONTAP.
- `ontap-nas-flexgroup`: Cada PV provisionado é um volume completo do Amazon FSX for NetApp ONTAP FlexGroup.

Para obter informações sobre o condutor, "Controladores NAS"consulte e "Controladores SAN".

Autenticação

O Astra Trident oferece dois modos de autenticação.

- Baseado em certificado: O Astra Trident se comunicará com o SVM em seu sistema de arquivos FSX usando um certificado instalado no seu SVM.
- Baseado em credenciais: Você pode usar o `fsxadmin` usuário para o sistema de arquivos ou o `vsadmin` usuário configurado para o SVM.

> O Astra Trident espera ser executado como um `vsadmin` usuário SVM ou como um usuário com um nome diferente que tenha a mesma função. O Amazon FSX for NetApp ONTAP tem um `fsxadmin` usuário que é uma substituição limitada do usuário do cluster do ONTAP `admin`. É altamente recomendável usar `vsadmin` com o Astra Trident.

Você pode atualizar backends para mover entre métodos baseados em credenciais e baseados em certificado. No entanto, se você tentar fornecer **credenciais e certificados**, a criação de backend falhará. Para alternar para um método de autenticação diferente, você deve remover o método existente da configuração de back-end.

Para obter detalhes sobre como ativar a autenticação, consulte a autenticação do tipo de driver:

- "Autenticação nas ONTAP"
- "Autenticação SAN ONTAP"

Identidade de nuvem para EKS

A identidade na nuvem permite que os pods do Kubernetes acessem recursos da AWS autenticando como uma função do AWS IAM em vez de fornecer credenciais explícitas da AWS.

Para aproveitar a identidade da nuvem na AWS, você precisa ter:

- Um cluster do Kubernetes implantado usando EKS
- Astra Trident instalado que inclui a `cloudProvider` especificação `"AWS"` e `cloudIdentity` especificação da função AWS IAM.

Operador Trident

Para instalar o Astra Trident usando o operador Trident, edite `tridentorchestrator_cr.yaml` para definir `cloudProvider` e definir `cloudIdentity` como `"AWS"` função AWS IAM.

Por exemplo:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/astratrident-role'"
```

Leme

Defina os valores para os sinalizadores **provedor de nuvem** e **identidade de nuvem** usando as seguintes variáveis de ambiente:

```
export CP="AWS"
export CI="'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/astratrident-role'"
```

O exemplo a seguir instala o Astra Trident e define `cloudProvider` como `AWS` usando a variável de ambiente `$CP` e define a 'cloudIdentity' usando a variável de ambiente `$CI` :

```
helm install trident trident-operator-100.2402.0.tgz --set
cloudProvider=$CP --set cloudIdentity=$CI
```

<code>dtridentctl</code>

Defina os valores para os sinalizadores **provedor de nuvem** e **identidade de nuvem** usando as seguintes variáveis de ambiente:

```
export CP="AWS"
export CI="'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/astratrident-role'"
```

O exemplo a seguir instala o Astra Trident e define o `cloud-provider` sinalizador como `$CP`, e `cloud-identity` como `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Encontre mais informações

- "Documentação do Amazon FSX para NetApp ONTAP"
- "Blog post no Amazon FSX for NetApp ONTAP"

Integre o Amazon FSX para NetApp ONTAP

Você pode integrar seu sistema de arquivos do Amazon FSX for NetApp ONTAP ao Astra Trident para garantir que os clusters do Kubernetes executados no Amazon Elastic Kubernetes Service (EKS) possam provisionar volumes persistentes de bloco e arquivo com o respaldo do ONTAP.

Requisitos

Além "Requisitos do Astra Trident"do , para integrar o FSX para ONTAP com Astra Trident, você precisa de:

- Um cluster do Amazon EKS existente ou um cluster do Kubernetes autogerenciado com `kubectl` o instalado.
- Um sistema de arquivos e máquina virtual de armazenamento (SVM) do Amazon FSX for NetApp ONTAP que pode ser acessado a partir dos nós de trabalho do seu cluster.
- Nós de trabalho preparados para "NFS ou iSCSI".

> Certifique-se de seguir as etapas de preparação de nós necessárias para o Amazon Linux e "Imagens de máquinas da Amazon" Ubuntu (AMIS), dependendo do seu tipo de AMI EKS.

- O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows. Prepare-se para provisionar volumes SMBConsulte para obter detalhes.

Integração de driver SAN e nas ONTAP

> Se você estiver configurando volumes SMB, leia Prepare-se para provisionar volumes SMB antes de criar o back-end.

Passos

1. Implante o Astra Trident com um dos "métodos de implantação".
2. Colete seu nome DNS de gerenciamento de SVM. Por exemplo, usando a AWS CLI, localize a `DNSName` entrada em `Endpoints` → `Management` depois de executar o seguinte comando:

```
aws fsx describe-storage-virtual-machines --region <file system region>
```

3. Criar e instalar certificados para "Autenticação de back-end nas" ou "Autenticação de back-end SAN".

Você pode fazer login no seu sistema de arquivos (por exemplo, para instalar certificados) usando SSH de qualquer lugar que possa chegar ao seu sistema de arquivos. Utilize o `fsxadmin` utilizador, a palavra-passe configurada quando criou o sistema de ficheiros e o nome DNS de gestão a partir `aws fsx describe-file-systems`do .

4. Crie um arquivo de back-end usando seus certificados e o nome DNS do seu LIF de gerenciamento, como mostrado na amostra abaixo:

YAML

```
version: 1
storageDriverName: ontap-san
backendName: customBackendName
managementLIF: svm-XXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXX.fsx.us-
east-2.aws.internal
svm: svm01
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "customBackendName",
  "managementLIF": "svm-XXXXXXXXXXXXXXXXX.fs-
XXXXXXXXXXXXXXXXX.fsx.us-east-2.aws.internal",
  "svm": "svm01",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz"
 }
```

Como alternativa, você pode criar um arquivo de back-end usando as credenciais SVM (nome de usuário e senha) armazenadas no AWS Secret Manager, conforme mostrado neste exemplo:

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  managementLIF:
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-
2:xxxxxxxx:secret:secret-name",
      "type": "awsarn"
    }
  }
}
```

Para obter informações sobre como criar backends, consulte estes links:

- "Configurar um back-end com drivers nas ONTAP"
- "Configure um back-end com drivers SAN ONTAP"

Prepare-se para provisionar volumes SMB

Você pode provisionar volumes SMB usando `ontap-nas` o driver. Antes de concluir Integração de driver SAN e nas ONTAPas etapas a seguir.

Antes de começar

Antes de provisionar volumes SMB usando `ontap-nas` o driver, você deve ter o seguinte:

- Um cluster do Kubernetes com um nó de controlador Linux e pelo menos um nó de trabalho do Windows que executa o Windows Server 2019. O Astra Trident é compatível com volumes SMB montados em pods executados apenas em nós do Windows.
- Pelo menos um segredo do Astra Trident que contém suas credenciais do ative Directory. Para gerar segredo `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Um proxy CSI configurado como um serviço Windows. Para configurar um `csi-proxy`, "GitHub: CSI Proxy"consulte ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes executados no Windows.

Passos

1. Criar compartilhamentos SMB. Você pode criar os compartilhamentos de administração SMB de duas maneiras usando o "Microsoft Management Console"snap-in pastas compartilhadas ou usando a CLI do ONTAP. Para criar compartilhamentos SMB usando a CLI do ONTAP:
   a. Se necessário, crie a estrutura do caminho do diretório para o compartilhamento.

      O `vserver cifs share create` comando verifica o caminho especificado na opção -path durante a criação de compartilhamento. Se o caminho especificado não existir, o comando falhará.

   b. Crie um compartilhamento SMB associado ao SVM especificado:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. Verifique se o compartilhamento foi criado:

```
vserver cifs share show -share-name share_name
```

      "Crie um compartilhamento SMB"Consulte para obter detalhes completos.

2. Ao criar o back-end, você deve configurar o seguinte para especificar volumes SMB. Para obter todas as opções de configuração de back-end do FSX for ONTAP, "Opções e exemplos de configuração do FSX for ONTAP"consulte .

Parâmetro	Descrição	Exemplo
`smbShare`	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP ou um nome para permitir que o Astra Trident crie o compartilhamento SMB. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP.	`smb-share`
`nasType`	**Tem de estar definido para `smb`.** Se nulo, o padrão é `nfs`.	`smb`
`securityStyle`	Estilo de segurança para novos volumes. **Deve ser definido como `ntfs` ou `mixed` para volumes SMB.**	`ntfs` Ou `mixed` para volumes SMB
`unixPermissions`	Modo para novos volumes. **Deve ser deixado vazio para volumes SMB.**	""

Opções e exemplos de configuração do FSX for ONTAP

Saiba mais sobre as opções de configuração de back-end para o Amazon FSX for ONTAP. Esta seção fornece exemplos de configuração de back-end.

Opções de configuração de back-end

Consulte a tabela a seguir para obter as opções de configuração de back-end:

Parâmetro	Descrição	Exemplo
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	`ontap-nas` `ontap-nas-economy`, , `ontap-nas-flexgroup` `ontap-san` , , , `ontap-san-economy`
`backendName`	Nome personalizado ou back-end de storage	Nome do driver

Parâmetro	Descrição	Exemplo
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	"10,0.0,1", "[2001:1234:abcd::fefe]"
`dataLIF`	Endereço IP do protocolo LIF. * ONTAP nas drivers*: Recomendamos especificar dataLIF. Se não for fornecido, o Astra Trident obtém LIFs de dados do SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) a ser usado para as operações de montagem NFS, permitindo que você crie um DNS de round-robin para balanceamento de carga em vários LIFs de dados. Pode ser alterado após a definição inicial. Consulte a . **Drivers SAN ONTAP**: Não especifique para iSCSI. O Astra Trident usa o mapa de LUN seletivo da ONTAP para descobrir as LIFs iSCI necessárias para estabelecer uma sessão de vários caminhos. Um aviso é gerado se o dataLIF for definido explicitamente. Pode ser definido para usar endereços IPv6 se o Astra Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	
`autoExportPolicy`	Ativar a criação e atualização automática da política de exportação [Boolean]. Com `autoExportPolicy` as opções e `autoExportCIDRs`, o Astra Trident pode gerenciar políticas de exportação automaticamente.	`false`

Parâmetro	Descrição	Exemplo
`autoExportCIDRs`	Lista de CIDR para filtrar IPs de nós do Kubernetes quando `autoExportPolicy` está ativado. Com `autoExportPolicy` as opções e `autoExportCIDRs`, o Astra Trident pode gerenciar políticas de exportação automaticamente.	"["0,0.0,0/0", ":::/0"]"
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""
`clientCertificate`	Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado	""
`clientPrivateKey`	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
`username`	Nome de usuário para se conetar ao cluster ou SVM. Usado para autenticação baseada em credenciais. Por exemplo, vsadmin.	
`password`	Senha para se conectar ao cluster ou SVM. Usado para autenticação baseada em credenciais.	
`svm`	Máquina virtual de armazenamento para usar	Derivado se um SVM managementLIF for especificado.
`storagePrefix`	Prefixo usado ao provisionar novos volumes na SVM. Não pode ser modificado após a criação. Para atualizar esse parâmetro, você precisará criar um novo backend.	`trident`
`limitAggregateUsage`	**Não especifique para o Amazon FSX for NetApp ONTAP.** O fornecido `fsxadmin` e `vsadmin` não contém as permissões necessárias para recuperar o uso agregado e limitá-lo usando o Astra Trident.	Não utilizar.

Parâmetro	Descrição	Exemplo
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor. Também restringe o tamanho máximo dos volumes que gerencia para qtrees e LUNs, e a `qtreesPerFlexvol` opção permite personalizar o número máximo de qtrees por FlexVol.	"" (não aplicado por padrão)
`lunsPerFlexvol`	O máximo de LUNs por FlexVol tem de estar no intervalo [50, 200]. Apenas SAN.	`100`
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Por exemplo, não use `debugTraceFlags` a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
`nfsMountOptions`	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes normalmente são especificadas em classes de storage, mas se nenhuma opção de montagem for especificada em uma classe de storage, o Astra Trident voltará a usar as opções de montagem especificadas no arquivo de configuração do back-end de storage. Se nenhuma opção de montagem for especificada na classe de storage ou no arquivo de configuração, o Astra Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb`, ou null. **Deve definir como `smb` para volumes SMB.** A configuração como null padrão para volumes NFS.	`nfs`
`qtreesPerFlexvol`	Qtrees máximos por FlexVol, têm de estar no intervalo [50, 300]	`200`

Parâmetro	Descrição	Exemplo
`smbShare`	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP ou um nome para permitir que o Astra Trident crie o compartilhamento SMB. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP.	`smb-share`
`useREST`	Parâmetro booleano para usar APIs REST do ONTAP. **A visualização técnica** `useREST` é fornecida como uma **prévia técnica** que é recomendada para ambientes de teste e não para cargas de trabalho de produção. Quando definido como `true`, o Astra Trident usará as APIS REST do ONTAP para se comunicar com o back-end. Esse recurso requer o ONTAP 9.11,1 e posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao `ontap` aplicativo. Isso é satisfeito com as funções e `cluster-admin` predefinidas `vsadmin`.	`false`
`aws`	Você pode especificar o seguinte no arquivo de configuração do AWS FSX for ONTAP: - `fsxFilesystemID`: Especifique o ID do sistema de arquivos AWS FSX. `apiRegion`- : Nome da região da API AWS. `apikey`- : Chave da API da AWS. `secretKey`- : Chave secreta da AWS.	`""` `""` `""`
`credentials`	Especifique as credenciais do FSX SVM para armazenar no AWS Secret Manager. `name`- : Nome do recurso Amazon (ARN) do segredo, que contém as credenciais do SVM. `type`- : Defina para `awsarn`. "Crie um segredo do AWS Secrets Manager"Consulte para obter mais informações.	

Atualização `dataLIF` após a configuração inicial

Você pode alterar o LIF de dados após a configuração inicial executando o seguinte comando para fornecer o novo arquivo JSON de back-end com LIF de dados atualizado.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

Se os PVCs estiverem anexados a um ou vários pods, você deverá reduzir todos os pods correspondentes e restaurá-los para que o novo LIF de dados entre em vigor.

Opções de configuração de back-end para volumes de provisionamento

Você pode controlar o provisionamento padrão usando essas opções na `defaults` seção da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para LUNs	`true`
`spaceReserve`	Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso)	`none`
`snapshotPolicy`	Política de instantâneos a utilizar	`none`
`qosPolicy`	Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento ou backend. O uso de grupos de política de QoS com o Astra Trident requer o ONTAP 9.8 ou posterior. Recomendamos o uso de um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de política de QoS compartilhado aplicará o limite máximo da taxa de transferência total de todos os workloads.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento ou backend. Não suportado pela ONTAP-nas-Economy.	""

Parâmetro	Descrição	Padrão
`snapshotReserve`	Porcentagem de volume reservado para snapshots "0"	Se `snapshotPolicy` for `none`, `else` ""
`splitOnClone`	Divida um clone de seu pai na criação	`false`
`encryption`	Ative a criptografia de volume do NetApp (NVE) no novo volume; o padrão é `false`. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se o NAE estiver ativado no back-end, qualquer volume provisionado no Astra Trident será o NAE ativado. Para obter mais informações, consulte: "Como o Astra Trident funciona com NVE e NAE".	`false`
`luksEncryption`	Ativar encriptação LUKS. "Usar a configuração de chave unificada do Linux (LUKS)"Consulte a . Apenas SAN.	""
`tieringPolicy`	Política de disposição em camadas para usar `none`	`snapshot-only` Para configuração pré-ONTAP 9.5 SVM-DR
`unixPermissions`	Modo para novos volumes. **Deixe vazio para volumes SMB.**	""
`securityStyle`	Estilo de segurança para novos volumes. Estilos de segurança e `unix` suporte de NFS `mixed`. Suporta SMB `mixed` e `ntfs` estilos de segurança.	O padrão NFS é `unix`. O padrão SMB é `ntfs`.

Exemplos de configurações

Configuração da classe de armazenamento para volumes SMB

Usando `nasType`, `node-stage-secret-name` e `node-stage-secret-namespace`, você pode especificar um volume SMB e fornecer as credenciais necessárias do ative Directory. Os volumes SMB são suportados usando `ontap-nas` apenas o driver.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nas-smb-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Configuração do AWS FSX for ONTAP com gerenciador de segredos

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  managementLIF:
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

Configure o complemento Astra Trident EKS versão 23,10 no cluster EKS

O Astra Trident simplifica o gerenciamento de armazenamento do Amazon FSX for NetApp ONTAP no Kubernetes para permitir que seus desenvolvedores e administradores se concentrem na implantação de aplicações. O complemento Astra Trident EKS inclui os patches de segurança mais recentes, correções de bugs e é validado pela AWS para funcionar com o Amazon EKS. O complemento EKS permite

que você garanta consistentemente que seus clusters do Amazon EKS estejam seguros e estáveis e reduza a quantidade de trabalho que você precisa fazer para instalar, configurar e atualizar complementos.

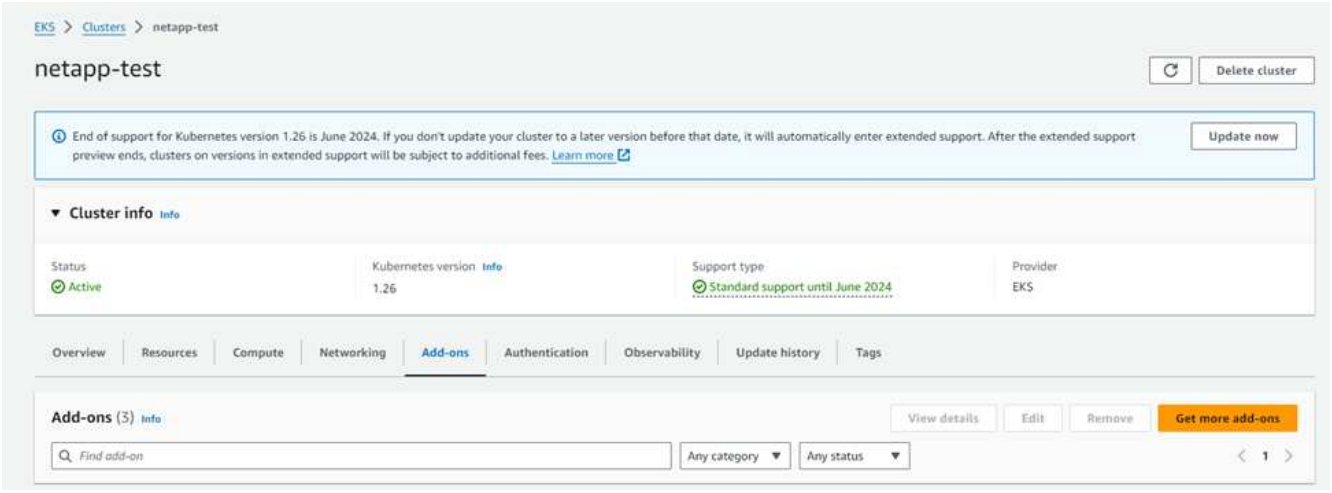
Pré-requisitos

Antes de configurar o complemento Astra Trident para AWS EKS, verifique se você tem o seguinte:

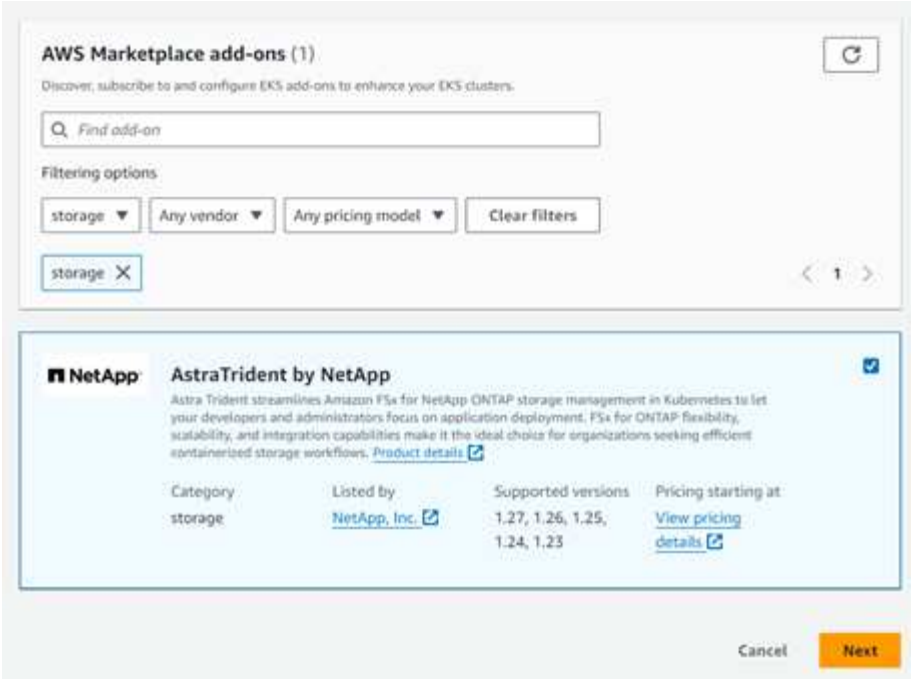
- Uma conta de cluster do Amazon EKS com assinatura complementar
- Permissões da AWS para o marketplace da AWS:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 ARM(AL2_ARM_64)
- Tipo de nó: AMD ou ARM
- Um sistema de arquivos existente do Amazon FSX for NetApp ONTAP

Passos

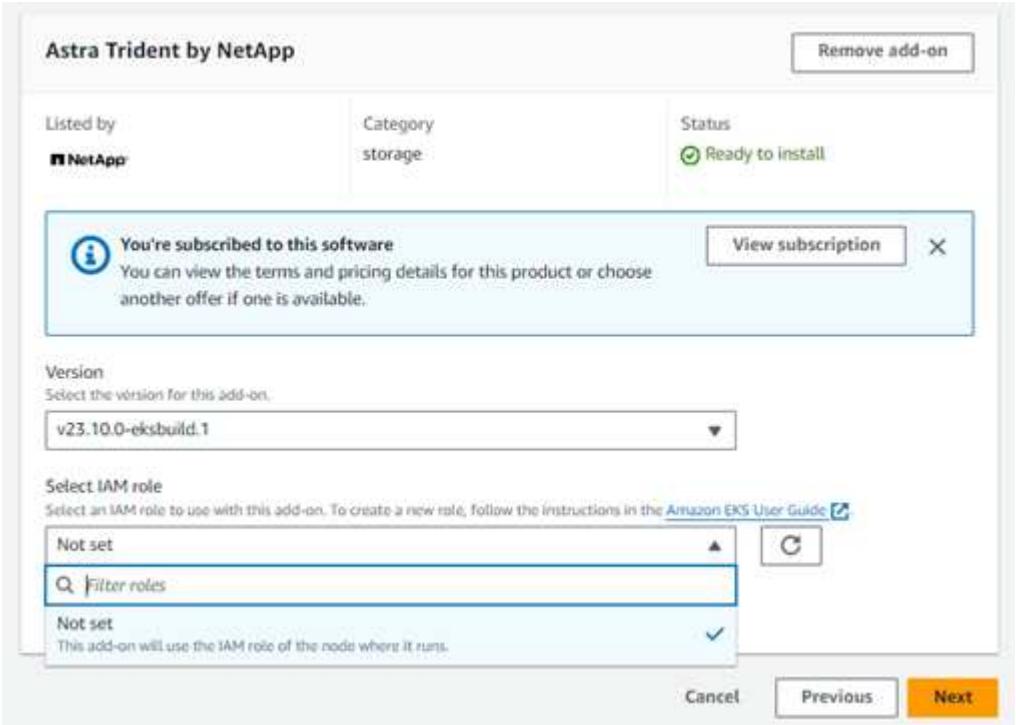
1. No cluster do EKS Kubernetes, navegue até a guia **Complementos**.



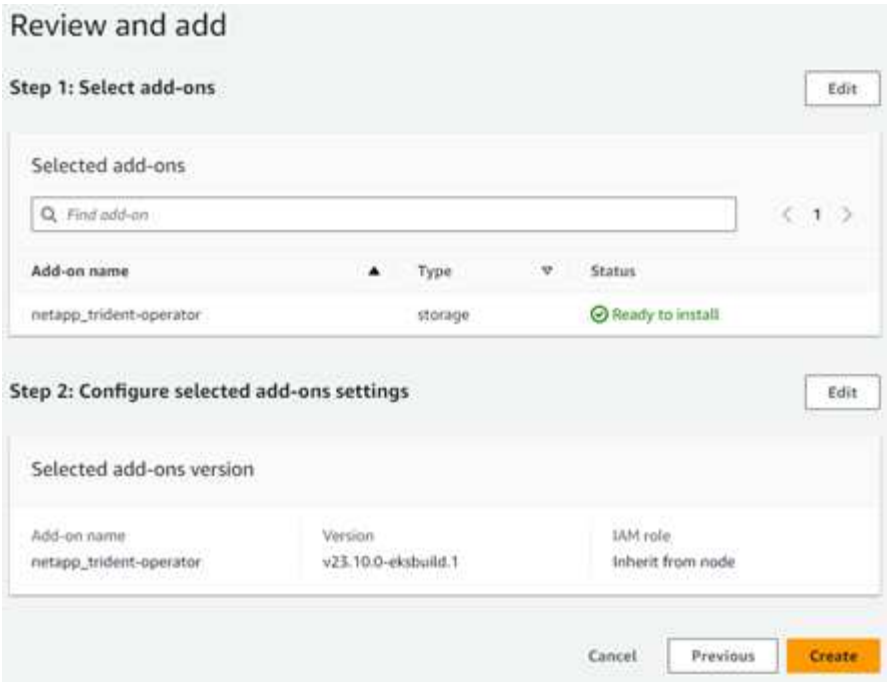
2. Vá para **Complementos do AWS Marketplace** e escolha a categoria *storage*.



3. Localize **AstraTrident by NetApp** e marque a caixa de seleção para o complemento Astra Trident.
4. Escolha a versão desejada do complemento.



5. Selecione a opção função do IAM para herdar do nó.
6. Configure quaisquer definições opcionais conforme necessário e selecione **seguinte**.



7. Selecione **criar**.
8. Verifique se o status do complemento é *ative*.

Instale/desinstale o complemento Astra Trident EKS usando a CLI

Instale o complemento Astra Trident EKS usando a CLI:

Os seguintes comandos de exemplo instalam o complemento Astra Trident EKS:
`eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version v23.10.0-eksbuild.`
`eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version v23.10.0-eksbuild.1` (Com uma versão dedicada)

Desinstale o complemento Astra Trident EKS usando a CLI:

O comando a seguir desinstala o complemento Astra Trident EKS:
`eksctl delete addon --cluster K8s-arm --name netapp_trident-operator`

Crie backends com kubectl

Um back-end define a relação entre o Astra Trident e um sistema de storage. Ele diz ao Astra Trident como se comunicar com esse sistema de storage e como o Astra Trident deve provisionar volumes a partir dele. Após a instalação do Astra Trident, a próxima etapa é criar um back-end. A `TridentBackendConfig` Definição de recursos personalizada (CRD) permite criar e gerenciar backends Trident diretamente por meio da

interface do Kubernetes. Para fazer isso, use `kubectl` ou a ferramenta CLI equivalente para sua distribuição do Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig`(`tbc tbconfig, , tbackendconfig`) É um CRD com namespaces e frontend que permite gerenciar backends Astra Trident usando `kubectl`o . Agora, os administradores de storage e Kubernetes podem criar e gerenciar back-ends diretamente pela CLI do Kubernetes sem exigir um utilitário de linha de comando dedicado (`tridentctl`).

Após a criação de `TridentBackendConfig` um objeto, acontece o seguinte:

- Um back-end é criado automaticamente pelo Astra Trident com base na configuração que você fornece. Isto é representado internamente como um `TridentBackend` (`tbe`, `tridentbackend`) CR.
- O `TridentBackendConfig` é exclusivamente vinculado a um `TridentBackend` que foi criado pelo Astra Trident.

Cada `TridentBackendConfig` um mantém um mapeamento um-para-um com um `TridentBackend`. o primeiro é a interface fornecida ao usuário para projetar e configurar backends; o último é como o Trident representa o objeto backend real.

> `TridentBackend` Os CRS são criados automaticamente pelo Astra Trident. Você **não deve** modificá-los. Se você quiser fazer atualizações para backends, faça isso modificando o `TridentBackendConfig` objeto.

Veja o exemplo a seguir para o formato do `TridentBackendConfig` CR:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Você também pode dar uma olhada nos exemplos "instalador do Trident" no diretório para configurações de exemplo para a plataforma/serviço de armazenamento desejado.

O `spec` utiliza parâmetros de configuração específicos do back-end. Neste exemplo, o backend usa o `ontap-san` driver de armazenamento e usa os parâmetros de configuração que são tabulados aqui. Para obter a lista de opções de configuração para o driver de armazenamento desejado, consulte o "informações de configuração de back-end para seu driver de armazenamento".

A `spec` seção também inclui `credentials` campos e `deletionPolicy`, que são recentemente introduzidos no `TridentBackendConfig` CR:

- `credentials`: Este parâmetro é um campo obrigatório e contém as credenciais usadas para autenticar com o sistema/serviço de armazenamento. Isso é definido como um segredo do Kubernetes criado pelo usuário. As credenciais não podem ser passadas em texto simples e resultarão em um erro.
- `deletionPolicy`: Este campo define o que deve acontecer quando o `TridentBackendConfig` é excluído. Pode tomar um dos dois valores possíveis:
  - `delete`: Isso resulta na exclusão do `TridentBackendConfig` CR e do back-end associado. Este é o valor padrão.
  - `retain`: Quando um `TridentBackendConfig` CR é excluído, a definição de back-end ainda estará presente e poderá ser gerenciada com `tridentctl`o . Definir a política de exclusão para `retain` permitir que os usuários façam o downgrade para uma versão anterior (anterior a 21,04) e mantenham os backends criados. O valor para este campo pode ser atualizado após a criação de um `TridentBackendConfig`.

O nome de um back-end é definido usando `spec.backendName`. Se não for especificado, o nome do backend é definido como o nome do `TridentBackendConfig` objeto (metadata.name). Recomenda-se definir explicitamente nomes de back-end usando `spec.backendName`o .

Backends que foram criados com `tridentctl` não têm um objeto associado `TridentBackendConfig`. Você pode optar por gerenciar esses backends `kubectl` criando um `TridentBackendConfig` CR. Deve-se ter cuidado para especificar parâmetros de configuração idênticos (como `spec.backendName` , , `spec.storagePrefix`, `spec.storageDriverName` e assim por diante). O Astra Trident vinculará automaticamente o recém-criado `TridentBackendConfig` ao back-end pré-existente.

Visão geral dos passos

Para criar um novo back-end usando `kubectl`o , você deve fazer o seguinte:

1. Criar um "Segredo do Kubernetes". o segredo contém as credenciais que o Astra Trident precisa para se comunicar com o cluster/serviço de storage.
2. Crie `TridentBackendConfig` um objeto. Isso contém detalhes sobre o cluster/serviço de armazenamento e faz referência ao segredo criado na etapa anterior.

Depois de criar um backend, você pode observar seu status usando `kubectl get tbc <tbc-name> -n <trident-namespace>` e coletar detalhes adicionais.

Etapa 1: Crie um segredo do Kubernetes

Crie um segredo que contenha as credenciais de acesso para o back-end. Isso é exclusivo para cada serviço/plataforma de storage. Aqui está um exemplo:

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: t@Ax@7q(>
```

Esta tabela resume os campos que devem ser incluídos no segredo para cada plataforma de armazenamento:

Descrição dos campos secretos da plataforma de armazenamento	Segredo	Descrição dos campos
Azure NetApp Files	ID do cliente	A ID do cliente a partir de um registo de aplicação
Cloud Volumes Service para GCP	private_key_id	ID da chave privada. Parte da chave da API para a conta de serviço do GCP com a função de administrador do CVS
Cloud Volumes Service para GCP	chave_privada	Chave privada. Parte da chave da API para a conta de serviço do GCP com a função de administrador do CVS
Elemento (NetApp HCI/SolidFire)	Endpoint	MVIP para o cluster SolidFire com credenciais de locatário
ONTAP	nome de utilizador	Nome de usuário para se conetar ao cluster/SVM. Usado para autenticação baseada em credenciais
ONTAP	palavra-passe	Senha para se conectar ao cluster/SVM. Usado para autenticação baseada em credenciais
ONTAP	ClientPrivateKey	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado

Descrição dos campos secretos da plataforma de armazenamento	Segredo	Descrição dos campos
ONTAP	ChapUsername	Nome de utilizador de entrada. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy`
ONTAP	IniciadorSecreto	Segredo do iniciador CHAP. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetUsername	Nome de utilizador alvo. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetInitiatorSecret	Segredo do iniciador de destino CHAP. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy`

O segredo criado nesta etapa será referenciado `spec.credentials` no campo do `TridentBackendConfig` objeto que é criado na próxima etapa.

Passo 2: Crie o `TridentBackendConfig` CR

Agora você está pronto para criar seu `TridentBackendConfig` CR. Neste exemplo, um back-end que usa `ontap-san` o driver é criado usando o `TridentBackendConfig` objeto mostrado abaixo:

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Etapa 3: Verifique o status do `TridentBackendConfig` CR

Agora que criou o `TridentBackendConfig` CR, pode verificar o estado. Veja o exemplo a seguir:

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                       BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

Um backend foi criado com sucesso e vinculado ao `TridentBackendConfig` CR.

A fase pode ter um dos seguintes valores:

- `Bound`: O `TridentBackendConfig` CR está associado a um back-end, e esse backend contém `configRef` definido como `TridentBackendConfig` uid do CR.
- `Unbound`: Representado `""` usando . O `TridentBackendConfig` objeto não está vinculado a um backend. Todos os CRS recém-criados `TridentBackendConfig` estão nesta fase por padrão. Após as alterações de fase, ela não pode voltar a Unbound.
- `Deleting`: Os `TridentBackendConfig` CR's `deletionPolicy` foram definidos para eliminar. Quando o `TridentBackendConfig` CR é excluído, ele passa para o estado de exclusão.
  - Se não houver declarações de volume persistentes (PVCs) no back-end, a exclusão do resultará na exclusão do `TridentBackendConfig` Astra Trident do back-end e do `TridentBackendConfig` CR.
  - Se um ou mais PVCs estiverem presentes no back-end, ele vai para um estado de exclusão. Posteriormente, o `TridentBackendConfig` CR entra também na fase de eliminação. O back-end e `TridentBackendConfig` são excluídos somente depois que todos os PVCs são excluídos.
- `Lost`: O back-end associado ao `TridentBackendConfig` CR foi acidentalmente ou deliberadamente excluído e o `TridentBackendConfig` CR ainda tem uma referência ao back-end excluído. O `TridentBackendConfig` CR ainda pode ser eliminado independentemente do `deletionPolicy` valor.

- `Unknown`: O Astra Trident não consegue determinar o estado ou a existência do back-end associado ao `TridentBackendConfig` CR. Por exemplo, se o servidor de API não estiver respondendo ou se o `tridentbackends.trident.netapp.io` CRD estiver ausente. Isso pode exigir intervenção.

Nesta fase, um backend é criado com sucesso! Existem várias operações que podem ser tratadas adicionalmente, "atualizações de back-end e exclusões de back-end"como o .

(Opcional) passo 4: Obtenha mais detalhes

Você pode executar o seguinte comando para obter mais informações sobre seu back-end:

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                      BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

Além disso, você também pode obter um despejo YAML/JSON do `TridentBackendConfig`.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: "2021-04-21T20:45:11Z"
  finalizers:
  - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` Contém o `backendName` e o `backendUUID` do back-end criado em resposta ao `TridentBackendConfig` CR. O `lastOperationStatus` campo representa o status da última operação `TridentBackendConfig` do CR, que pode ser acionada pelo usuário (por exemplo, o usuário mudou algo no `spec`) ou acionada pelo Astra Trident (por exemplo, durante reinicializações do Astra Trident). Pode ser sucesso ou falhou. `phase` Representa o status da relação entre o `TridentBackendConfig` CR e o back-end. No exemplo acima, `phase` tem o valor vinculado, o que significa que o `TridentBackendConfig` CR está associado ao back-end.

Você pode executar o `kubectl -n trident describe tbc <tbc-cr-name>` comando para obter detalhes dos logs de eventos.

> Não é possível atualizar ou excluir um back-end que contenha um objeto `tridentctl` associado `TridentBackendConfig` usando o . Compreender as etapas envolvidas na troca entre `tridentctl` e `TridentBackendConfig`, "veja aqui".

Gerenciar backends

Execute o gerenciamento de back-end com o kubectl

Saiba mais sobre como executar operações de gerenciamento de back-end usando `kubectl`o .

Excluir um back-end

Ao excluir um `TridentBackendConfig`, você instrui o Astra Trident a excluir/reter backends (com base `deletionPolicy` no). Para excluir um back-end, certifique-se de que `deletionPolicy` está definido para excluir. Para eliminar apenas o `TridentBackendConfig`, certifique-se de que `deletionPolicy` está definido como reter. Isso garantirá que o backend ainda esteja presente e possa ser gerenciado usando `tridentctl`o .

Execute o seguinte comando:

```
kubectl delete tbc <tbc-name> -n trident
```

O Astra Trident não exclui os segredos do Kubernetes que estavam em uso `TridentBackendConfig` pelo . O usuário do Kubernetes é responsável pela limpeza de segredos. Cuidado deve ser tomado ao excluir segredos. Você deve excluir segredos somente se eles não estiverem em uso pelos backends.

Veja os backends existentes

Execute o seguinte comando:

```
kubectl get tbc -n trident
```

Você também pode executar `tridentctl get backend -n trident` ou `tridentctl get backend -o yaml -n trident` obter uma lista de todos os backends que existem. Esta lista também incluirá backends que foram criados com `tridentctl`.

Atualize um back-end

Pode haver várias razões para atualizar um backend:

- As credenciais para o sistema de storage foram alteradas. Para atualizar as credenciais, o segredo do Kubernetes que é usado no `TridentBackendConfig` objeto deve ser atualizado. O Astra Trident atualizará automaticamente o back-end com as credenciais mais recentes fornecidas. Execute o seguinte comando para atualizar o segredo do Kubernetes:

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- Os parâmetros (como o nome do SVM do ONTAP sendo usado) precisam ser atualizados.
  - Você pode atualizar `TridentBackendConfig` objetos diretamente pelo Kubernetes usando o seguinte comando:

```
kubectl apply -f <updated-backend-file.yaml>
```

- Alternativamente, você pode fazer alterações no CR existente `TridentBackendConfig` usando o seguinte comando:

```
kubectl edit tbc <tbc-name> -n trident
```

- Se uma atualização de back-end falhar, o back-end continuará em sua última configuração conhecida. Pode visualizar os registos para determinar a causa executando `kubectl get tbc <tbc-name> -o yaml -n trident` ou `kubectl describe tbc <tbc-name> -n trident`.
- Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar novamente o comando update.

Execute o gerenciamento de back-end com o tridentctl

Saiba mais sobre como executar operações de gerenciamento de back-end usando `tridentctl`o .

Crie um backend

Depois de criar um "arquivo de configuração de back-end", execute o seguinte comando:

```
tridentctl create backend -f <backend-file> -n trident
```

Se a criação do backend falhar, algo estava errado com a configuração do backend. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs -n trident
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode simplesmente executar o `create` comando novamente.

Excluir um back-end

Para excluir um back-end do Astra Trident, faça o seguinte:

1. Recuperar o nome do backend:

```
tridentctl get backend -n trident
```

2. Excluir o backend:

```
tridentctl delete backend <backend-name> -n trident
```

Se o Astra Trident provisionou volumes e snapshots desse back-end que ainda existem, a exclusão do back-end impede que novos volumes sejam provisionados por ele. O back-end continuará a existir em um estado de exclusão e o Trident continuará a gerenciar esses volumes e snapshots até que sejam excluídos.

Veja os backends existentes

Para visualizar os backends que o Trident conhece, faça o seguinte:

- Para obter um resumo, execute o seguinte comando:

```
tridentctl get backend -n trident
```

- Para obter todos os detalhes, execute o seguinte comando:

```
tridentctl get backend -o json -n trident
```

Atualize um back-end

Depois de criar um novo arquivo de configuração de back-end, execute o seguinte comando:

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Se a atualização do backend falhar, algo estava errado com a configuração do backend ou você tentou uma atualização inválida. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs -n trident
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode simplesmente executar o `update` comando novamente.

Identificar as classes de armazenamento que usam um back-end

Este é um exemplo do tipo de perguntas que você pode responder com o JSON que `tridentctl` produz para objetos de back-end. Isso usa o `jq` utilitário, que você precisa instalar.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Isso também se aplica a backends que foram criados usando `TridentBackendConfig`o .

Alternar entre opções de gerenciamento de back-end

Saiba mais sobre as diferentes maneiras de gerenciar back-ends no Astra Trident.

Opções para gerenciar backends

Com a introdução `TridentBackendConfig`do , os administradores agora têm duas maneiras exclusivas de gerenciar backends. Isso coloca as seguintes perguntas:

- Os backends podem ser criados usando `tridentctl` ser gerenciados com `TridentBackendConfig`?
- Os backends podem ser criados usando `TridentBackendConfig` ser gerenciados `tridentctl` usando ?

Gerenciar `tridentctl` backends usando `TridentBackendConfig`

Esta seção aborda as etapas necessárias para gerenciar backends que foram criados usando `tridentctl` diretamente a interface do Kubernetes criando `TridentBackendConfig` objetos.

Isso se aplicará aos seguintes cenários:

- Backends pré-existentes, que não têm um `TridentBackendConfig` porque foram criados com `tridentctl`.
- Novos backends que foram criados com `tridentctl`, enquanto outros `TridentBackendConfig` objetos existem.

Em ambos os cenários, os back-ends continuarão presentes, com o Astra Trident agendando volumes e operando neles. Os administradores têm uma das duas opções aqui:

- Continue `tridentctl` usando para gerenciar backends que foram criados usando-o.
- Vincular backends criados usando `tridentctl` a um novo `TridentBackendConfig` objeto. Fazer isso significaria que os backends serão gerenciados usando `kubectl` e não `tridentctl`.

Para gerenciar um back-end pré-existente usando `kubectl`o , você precisará criar um `TridentBackendConfig` que se vincule ao back-end existente. Aqui está uma visão geral de como isso funciona:

1. Crie um segredo do Kubernetes. O segredo contém as credenciais que o Astra Trident precisa para se comunicar com o cluster/serviço de storage.
2. Crie `TridentBackendConfig` um objeto. Isso contém detalhes sobre o cluster/serviço de armazenamento e faz referência ao segredo criado na etapa anterior. Deve-se ter cuidado para especificar parâmetros de configuração idênticos (como `spec.backendName` , , `spec.storagePrefix`, `spec.storageDriverName` e assim por diante). `spec.backendName` deve ser definido como o nome do backend existente.

Passo 0: Identifique o backend

Para criar um `TridentBackendConfig` que se vincula a um backend existente, você precisará obter a configuração de backend. Neste exemplo, vamos supor que um backend foi criado usando a seguinte definição JSON:

```
tridentctl get backend ontap-nas-backend -n trident
```

```
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+

cat ontap-nas-backend.json

{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.10.10.1",
    "dataLIF": "10.10.10.2",
    "backendName": "ontap-nas-backend",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "admin-password",

    "defaults": {
        "spaceReserve": "none",
        "encryption": "false"
    },
    "labels":{"store":"nas_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
```

```
            }
        }
    ]
}
```

Etapa 1: Crie um segredo do Kubernetes

Crie um segredo que contenha as credenciais para o back-end, como mostrado neste exemplo:

```
cat tbc-ontap-nas-backend-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password

kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

Passo 2: Crie um `TridentBackendConfig` CR

O próximo passo é criar um `TridentBackendConfig` CR que se vinculará automaticamente ao pré-existente `ontap-nas-backend` (como neste exemplo). Certifique-se de que os seguintes requisitos são cumpridos:

- O mesmo nome de back-end é definido no `spec.backendName`.
- Os parâmetros de configuração são idênticos ao back-end original.
- Os pools virtuais (se presentes) devem manter a mesma ordem que no back-end original.
- As credenciais são fornecidas por meio de um segredo do Kubernetes e não em texto simples.

Neste caso, o `TridentBackendConfig` será parecido com este:

```
cat backend-tbc-ontap-nas.yaml
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'

kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

Etapa 3: Verifique o status do `TridentBackendConfig` CR

Após a criação do `TridentBackendConfig` , sua fase deve ser `Bound`. Ele também deve refletir o mesmo nome de back-end e UUID que o do back-end existente.

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                   BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

O backend agora será completamente gerenciado usando o `tbc-ontap-nas-backend` `TridentBackendConfig` objeto.

Gerenciar `TridentBackendConfig` backends usando `tridentctl`

```
`tridentctl` pode ser usado para listar backends que foram criados usando
`TridentBackendConfig`. Além disso, os administradores também podem optar
por gerenciar completamente esses backends `tridentctl` excluindo
`TridentBackendConfig` e certificando-se de `spec.deletionPolicy` que está
definido como `retain`.
```

Passo 0: Identifique o backend

Por exemplo, vamos supor que o seguinte backend foi criado usando `TridentBackendConfig`:

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

A partir da saída, vê-se que `TridentBackendConfig` foi criado com sucesso e está vinculado a um backend [observe o UUID do backend].

Passo 1: Confirmar `deletionPolicy` **está definido como** `retain`

Vamos dar uma olhada no valor `deletionPolicy` de . Isso precisa ser definido como `retain`. Isso garantirá que, quando um `TridentBackendConfig` CR for excluído, a definição de back-end ainda estará presente e poderá ser gerenciada com `tridentctl`o .

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

Não avance para o passo seguinte, a menos `deletionPolicy` que esteja definido para `retain`.

Etapa 2: Exclua o `TridentBackendConfig` CR

O passo final é eliminar o `TridentBackendConfig` CR. Depois de confirmar que o `deletionPolicy` está definido como `retain`, pode avançar com a eliminação:

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE   | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Após a exclusão `TridentBackendConfig` do objeto, o Astra Trident simplesmente o remove sem realmente excluir o próprio back-end.

Criar e gerenciar classes de armazenamento

Crie uma classe de armazenamento

Configure um objeto Kubernetes StorageClass e crie a classe de storage para instruir o Astra Trident a provisionar volumes.

Configurar um objeto Kubernetes StorageClass

O "Objeto Kubernetes StorageClass" identifica o Astra Trident como o provisionador que é usado para essa classe e instrui o Astra Trident a provisionar um volume. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

"Objetos Kubernetes e Trident"Consulte para obter detalhes sobre como as classes de storage interagem com os `PersistentVolumeClaim` parâmetros e para controlar como o Astra Trident provisiona volumes.

Crie uma classe de armazenamento

Depois de criar o objeto StorageClass, você pode criar a classe de armazenamento. Amostras da classe de armazenamento fornece algumas amostras básicas que você pode usar ou modificar.

Passos

1. Esse é um objeto do Kubernetes, então use `kubectl` para criá-lo no Kubernetes.

   ```
   kubectl create -f sample-input/storage-class-basic-csi.yaml
   ```

2. Agora você deve ver uma classe de storage **Basic-csi** no Kubernetes e Astra Trident, e o Astra Trident deve ter descoberto os pools no back-end.

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Amostras da classe de armazenamento

O Astra Trident "definições de classe de armazenamento simples para backends específicos" fornece .

Alternativamente, você pode editar `sample-input/storage-class-csi.yaml.templ` o arquivo que vem com o instalador e substituir *`BACKEND_TYPE`* pelo nome do driver de armazenamento.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Gerenciar classes de armazenamento

Você pode exibir classes de armazenamento existentes, definir uma classe de armazenamento padrão, identificar o back-end da classe de armazenamento e excluir classes de armazenamento.

Exibir as classes de armazenamento existentes

- Para visualizar as classes de armazenamento do Kubernetes existentes, execute o seguinte comando:

```
kubectl get storageclass
```

- Para ver os detalhes da classe de storage do Kubernetes, execute o seguinte comando:

```
kubectl get storageclass <storage-class> -o json
```

- Para exibir as classes de storage sincronizadas do Astra Trident, execute o seguinte comando:

```
tridentctl get storageclass
```

- Para visualizar os detalhes da classe de storage sincronizado do Astra Trident, execute o seguinte comando:

```
tridentctl get storageclass <storage-class> -o json
```

Defina uma classe de armazenamento padrão

O Kubernetes 1,6 adicionou a capacidade de definir uma classe de storage padrão. Esta é a classe de armazenamento que será usada para provisionar um volume persistente se um usuário não especificar um em uma reivindicação de volume persistente (PVC).

- Defina uma classe de armazenamento padrão definindo a anotação `storageclass.kubernetes.io/is-default-class` como verdadeira na definição da classe de armazenamento. De acordo com a especificação, qualquer outro valor ou ausência da anotação é interpretado como falso.
- Você pode configurar uma classe de armazenamento existente para ser a classe de armazenamento padrão usando o seguinte comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- Da mesma forma, você pode remover a anotação de classe de armazenamento padrão usando o seguinte comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Há também exemplos no pacote de instalação do Trident que incluem esta anotação.

> Deve haver apenas uma classe de armazenamento padrão no cluster de cada vez. O Kubernetes não impede tecnicamente que você tenha mais de um, mas se comportará como se não houvesse nenhuma classe de storage padrão.

Identificar o back-end de uma classe de storage

Este é um exemplo do tipo de perguntas que você pode responder com o JSON que `tridentctl` produz para objetos backend Astra Trident. Isso usa o `jq` utilitário, que você pode precisar instalar primeiro.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

Excluir uma classe de armazenamento

Para excluir uma classe de armazenamento do Kubernetes, execute o seguinte comando:

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` deve ser substituído pela sua classe de armazenamento.

Todos os volumes persistentes criados com essa classe de storage permanecerão intocados, e o Astra Trident continuará gerenciá-los.

> O Astra Trident impõe um espaço em branco `fsType` para os volumes que cria. Para backends iSCSI, recomenda-se aplicar `parameters.fsType` no StorageClass. Você deve excluir StorageClasses existentes e recriá-los com `parameters.fsType` especificado.

Provisionar e gerenciar volumes

Provisionar um volume

Crie um Persistentvolume (PV) e um PersistentVolumeClaim (PVC) que use o Kubernetes StorageClass configurado para solicitar acesso ao PV. Em seguida, pode montar o PV num pod.

Visão geral

A "*Persistentvolume*" (PV) é um recurso de armazenamento físico provisionado pelo administrador de cluster em um cluster do Kubernetes. O "*PersistentVolumeClaim*" (PVC) é um pedido de acesso ao Persistentvolume no cluster.

O PVC pode ser configurado para solicitar o armazenamento de um determinado tamanho ou modo de acesso. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que o Persistentvolume e o modo de acesso, como desempenho ou nível de serviço.

Depois de criar o PV e o PVC, você pode montar o volume em um pod.

Manifestos de amostra

Persistentvolume Sample MANIFEST

Este manifesto de exemplo mostra um PV básico de 10Gi que está associado ao StorageClass . `basic-csi`

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

PersistentVolumeClaim amostra manifestos

Estes exemplos mostram opções básicas de configuração de PVC.

PVC com acesso RWO

Este exemplo mostra um PVC básico com acesso RWO associado a um StorageClass `basic-csi` chamado .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC com NVMe/TCP

Este exemplo mostra um PVC básico para NVMe/TCP com acesso RWO associado a um StorageClass `protection-gold` chamado .

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Amostras de manifesto POD

Estes exemplos mostram configurações básicas para anexar o PVC a um pod.

Configuração básica

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

Configuração básica NVMe/TCP

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      resources: {}
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
      claimName: pvc-san-nvme
```

Crie o PV e o PVC

Passos

1. Crie o PV.

```
kubectl create -f pv.yaml
```

2. Verifique o estado do PV.

```
kubectl get pv
NAME        CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM
STORAGECLASS  REASON  AGE
pv-storage  4Gi       RWO           Retain          Available
7s
```

3. Crie o PVC.

```
kubectl create -f pvc.yaml
```

4. Verifique o estado do PVC.

```
kubectl get pvc
NAME        STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi      RWO                       5m
```

5. Monte o volume num pod.

```
kubectl create -f pv-pod.yaml
```

> Pode monitorizar o progresso utilizando `kubectl get pod --watch`o .

6. Verifique se o volume está montado no `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

7. Agora você pode excluir o Pod. O aplicativo Pod não existirá mais, mas o volume permanecerá.

```
kubectl delete pod task-pv-pod
```

"Objetos Kubernetes e Trident"Consulte para obter detalhes sobre como as classes de storage interagem com os `PersistentVolumeClaim` parâmetros e para controlar como o Astra Trident provisiona volumes.

Expanda volumes

O Astra Trident oferece aos usuários do Kubernetes a capacidade de expandir seus volumes depois que eles são criados. Encontre informações sobre as configurações necessárias para expandir volumes iSCSI e NFS.

Expanda um volume iSCSI

É possível expandir um iSCSI Persistent volume (PV) usando o provisionador de CSI.

> A expansão de volume iSCSI é suportada pelos `ontap-san` `ontap-san-economy` drivers , , `solidfire-san` e requer o Kubernetes 1,16 e posterior.

Etapa 1: Configure o StorageClass para dar suporte à expansão de volume

Edite a definição StorageClass para definir o `allowVolumeExpansion` campo como `true`.

```
cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Para um StorageClass já existente, edite-o para incluir o `allowVolumeExpansion` parâmetro.

Etapa 2: Crie um PVC com o StorageClass que você criou

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o tamanho recém-desejado, que deve ser maior do que o tamanho original.

```
cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

O Astra Trident cria um volume persistente (PV) e o associa a essa reivindicação de volume persistente (PVC).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Passo 3: Defina um pod que prende o PVC

Fixe o PV a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um iSCSI PV:

- Se o PV estiver conetado a um pod, o Astra Trident expande o volume no back-end de armazenamento, refaz o dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, o Astra Trident expande o volume no back-end de armazenamento. Depois que o PVC é ligado a um pod, o Trident refaz o dispositivo e redimensiona o sistema de arquivos. Em seguida, o Kubernetes atualiza o tamanho do PVC após a operação de expansão ter sido concluída com sucesso.

Neste exemplo, é criado um pod que usa o `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Passo 4: Expanda o PV

Para redimensionar o PV que foi criado de 1Gi a 2Gi, edite a definição de PVC e atualize o `spec.resources.requests.storage` para 2Gi.

```
kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 ...
```

Etapa 5: Validar a expansão

É possível validar a expansão trabalhada corretamente verificando o tamanho do PVC, PV e volume Astra Trident:

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Expandir um volume NFS

O Astra Trident dá suporte à expansão de volume para PVS NFS provisionados em `ontap-nas` `ontap-nas-economy` , , , `ontap-nas-flexgroup` `gcp-cvs` e `azure-netapp-files` backends.

Etapa 1: Configure o StorageClass para dar suporte à expansão de volume

Para redimensionar um PV NFS, o administrador primeiro precisa configurar a classe de armazenamento para permitir a expansão de volume definindo o `allowVolumeExpansion` campo para `true`:

```
cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Se você já criou uma classe de armazenamento sem essa opção, você pode simplesmente editar a classe de armazenamento existente usando `kubectl edit storageclass` para permitir a expansão de volume.

Etapa 2: Crie um PVC com o StorageClass que você criou

```
cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

O Astra Trident deve criar um PV NFS de 20MiB para este PVC:

```
kubectl get pvc
NAME           STATUS   VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

Passo 3: Expanda o PV

Para redimensionar o 20MiB PV recém-criado para 1GiB, edite o PVC e defina `spec.resources.requests.storage` como 1GiB:

```
kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
...
```

Etapa 4: Validar a expansão

Você pode validar o redimensionamento trabalhado corretamente verificando o tamanho do PVC, PV e o volume Astra Trident:

```
kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Importar volumes

Você pode importar volumes de armazenamento existentes como um PV do Kubernetes usando `tridentctl import`o .

Visão geral e considerações

Você pode importar um volume para o Astra Trident para:

- Conteinerize um aplicativo e reutilize seu conjunto de dados existente
- Use um clone de um conjunto de dados para uma aplicação efêmera
- Reconstruir um cluster do Kubernetes com falha
- Migrar dados da aplicação durante a recuperação de desastre

Considerações

Antes de importar um volume, reveja as seguintes considerações.

- O Astra Trident pode importar apenas volumes ONTAP do tipo RW (leitura-gravação). Os volumes do tipo DP (proteção de dados) são volumes de destino do SnapMirror. Você deve quebrar a relação espelhada antes de importar o volume para o Astra Trident.

- Sugerimos importar volumes sem conexões ativas. Para importar um volume usado ativamente, clonar o volume e, em seguida, executar a importação.

Isso é especialmente importante para volumes de bloco, já que o Kubernetes não sabia da conexão anterior e poderia facilmente anexar um volume ativo a um pod. Isso pode resultar em corrupção de dados.

- Embora `StorageClass` deva ser especificado em um PVC, o Astra Trident não usa esse parâmetro durante a importação. As classes de armazenamento são usadas durante a criação de volume para selecionar os pools disponíveis com base nas caraterísticas de armazenamento. Como o volume já existe, nenhuma seleção de pool é necessária durante a importação. Portanto, a importação não falhará mesmo se o volume existir em um backend ou pool que não corresponda à classe de armazenamento especificada no PVC.
- O tamanho do volume existente é determinado e definido no PVC. Depois que o volume é importado pelo driver de armazenamento, o PV é criado com uma ClaimRef para o PVC.
  - A política de recuperação é inicialmente definida como `retain` no PV. Depois que o Kubernetes vincula com êxito o PVC e o PV, a política de recuperação é atualizada para corresponder à política de recuperação da Classe de armazenamento.
  - Se a política de recuperação da Classe de armazenamento for `delete`, o volume de armazenamento será excluído quando o PV for excluído.
- Por padrão, o Astra Trident gerencia o PVC e renomeia o FlexVol e o LUN no back-end. Você pode passar o `--no-manage` sinalizador para importar um volume não gerenciado. Se você usar `--no-manage`o , o Astra Trident não realiza nenhuma operação adicional no PVC ou no PV para o ciclo de vida dos objetos. O volume de armazenamento não é excluído quando o PV é excluído e outras operações, como clone de volume e redimensionamento de volume também são ignoradas.

Essa opção é útil se você quiser usar o Kubernetes para workloads em contêineres, mas de outra forma quiser gerenciar o ciclo de vida do volume de storage fora do Kubernetes.

- Uma anotação é adicionada ao PVC e ao PV que serve para um duplo propósito de indicar que o volume foi importado e se o PVC e o PV são gerenciados. Esta anotação não deve ser modificada ou removida.

Importar um volume

Pode utilizar `tridentctl import` para importar um volume.

Passos

1. Crie o arquivo PVC (Persistent volume Claim) (por exemplo, `pvc.yaml`) que será usado para criar o PVC. O ficheiro PVC deve incluir `name` `namespace` , , , `accessModes` `storageClassName` e . Opcionalmente, você pode especificar `unixPermissions` em sua definição de PVC.

   O seguinte é um exemplo de uma especificação mínima:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

Não inclua parâmetros adicionais, como nome PV ou tamanho do volume. Isso pode fazer com que o comando de importação falhe.

2. Use o `tridentctl import` comando para especificar o nome do back-end do Astra Trident que contém o volume e o nome que identifica exclusivamente o volume no storage (por exemplo: ONTAP FlexVol, Element volume, caminho Cloud Volumes Service). O `-f` argumento é necessário para especificar o caminho para o arquivo PVC.

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
```

Exemplos

Reveja os exemplos de importação de volume a seguir para drivers suportados.

ONTAP nas e ONTAP nas FlexGroup

O Astra Trident é compatível com a importação de volume usando `ontap-nas` os drivers e. `ontap-nas-flexgroup`

- O `ontap-nas-economy` driver não pode importar e gerenciar qtrees.
- Os `ontap-nas` drivers e `ontap-nas-flexgroup` não permitem nomes de volume duplicados.

Cada volume criado com o `ontap-nas` driver é um FlexVol no cluster do ONTAP. A importação do FlexVols com o `ontap-nas` driver funciona da mesma forma. Um FlexVol que já existe em um cluster ONTAP pode ser importado como `ontap-nas` PVC. Da mesma forma, os vols FlexGroup podem ser importados como `ontap-nas-flexgroup` PVCs.

Exemplos de ONTAP nas

A seguir mostra um exemplo de um volume gerenciado e uma importação de volume não gerenciado.

Volume gerenciado

O exemplo a seguir importa um volume nomeado `managed_volume` em um backend chamado `ontap_nas`:

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
```

Volume não gerenciado

Ao usar o `--no-manage` argumento, o Astra Trident não renomeará o volume.

O exemplo a seguir é importado `unmanaged_volume` `ontap_nas` no backend:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
```

San ONTAP

O Astra Trident é compatível com a importação de volume usando `ontap-san` o driver. A importação de volume não é suportada usando `ontap-san-economy` o driver.

O Astra Trident pode importar ONTAP SAN FlexVols que contenham um único LUN. Isso é consistente com o `ontap-san` driver, que cria um FlexVol para cada PVC e um LUN dentro do FlexVol. O Astra Trident importa o FlexVol e associa-o à definição de PVC.

Exemplos de SAN ONTAP

A seguir mostra um exemplo de um volume gerenciado e uma importação de volume não gerenciado.

Volume gerenciado

Para volumes gerenciados, o Astra Trident renomeia o FlexVol para `pvc-<uuid>` o formato e o LUN no FlexVol para `lun0`.

O exemplo a seguir importa `ontap-san-managed` o FlexVol que está presente no `ontap_san_default` back-end:

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Volume não gerenciado

O exemplo a seguir é importado `unmanaged_example_volume` `ontap_san` no backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Se você tiver LUNS mapeados para grupos que compartilham uma IQN com um nó Kubernetes IQN, como mostrado no exemplo a seguir, você receberá o erro: `LUN already mapped to initiator(s) in this group`. Você precisará remover o iniciador ou desmapear o LUN para importar o volume.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

Elemento

O Astra Trident é compatível com o software NetApp Element e a importação de volume NetApp HCI usando `solidfire-san` o driver.

> ⓘ O driver Element suporta nomes de volume duplicados. No entanto, o Astra Trident retorna um erro se houver nomes de volume duplicados. Como solução alternativa, clone o volume, forneça um nome de volume exclusivo e importe o volume clonado.

Exemplo de elemento

O exemplo a seguir importa um `element-managed` volume no backend . `element_default`

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud Platform

O Astra Trident é compatível com a importação de volume usando `gcp-cvs` o driver.

> ⓘ Para importar um volume com o suporte do NetApp Cloud Volumes Service no Google Cloud Platform, identifique o volume pelo caminho de volume. O caminho do volume é a parte do caminho de exportação do volume após o `:/`. Por exemplo, se o caminho de exportação for `10.0.0.1:/adroit-jolly-swift`, o caminho do volume será `adroit-jolly-swift`.

Exemplo do Google Cloud Platform

O exemplo a seguir importa um `gcp-cvs` volume no back-end `gcpcvs_YEppr` com o caminho de volume `adroit-jolly-swift` do .

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-
file> -n trident

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                  NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

O Astra Trident é compatível com a importação de volume usando `azure-netapp-files` o driver.

> Para importar um volume Azure NetApp Files, identifique o volume pelo seu caminho de volume. O caminho do volume é a parte do caminho de exportação do volume após o `:/`. Por exemplo, se o caminho de montagem for `10.0.0.2:/importvol1`, o caminho do volume será `importvol1`.

Exemplo de Azure NetApp Files

O exemplo a seguir importa um `azure-netapp-files` volume no back-end `azurenetappfiles_40517` com o caminho do volume `importvol1` .

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                  NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Compartilhar um volume NFS entre namespaces

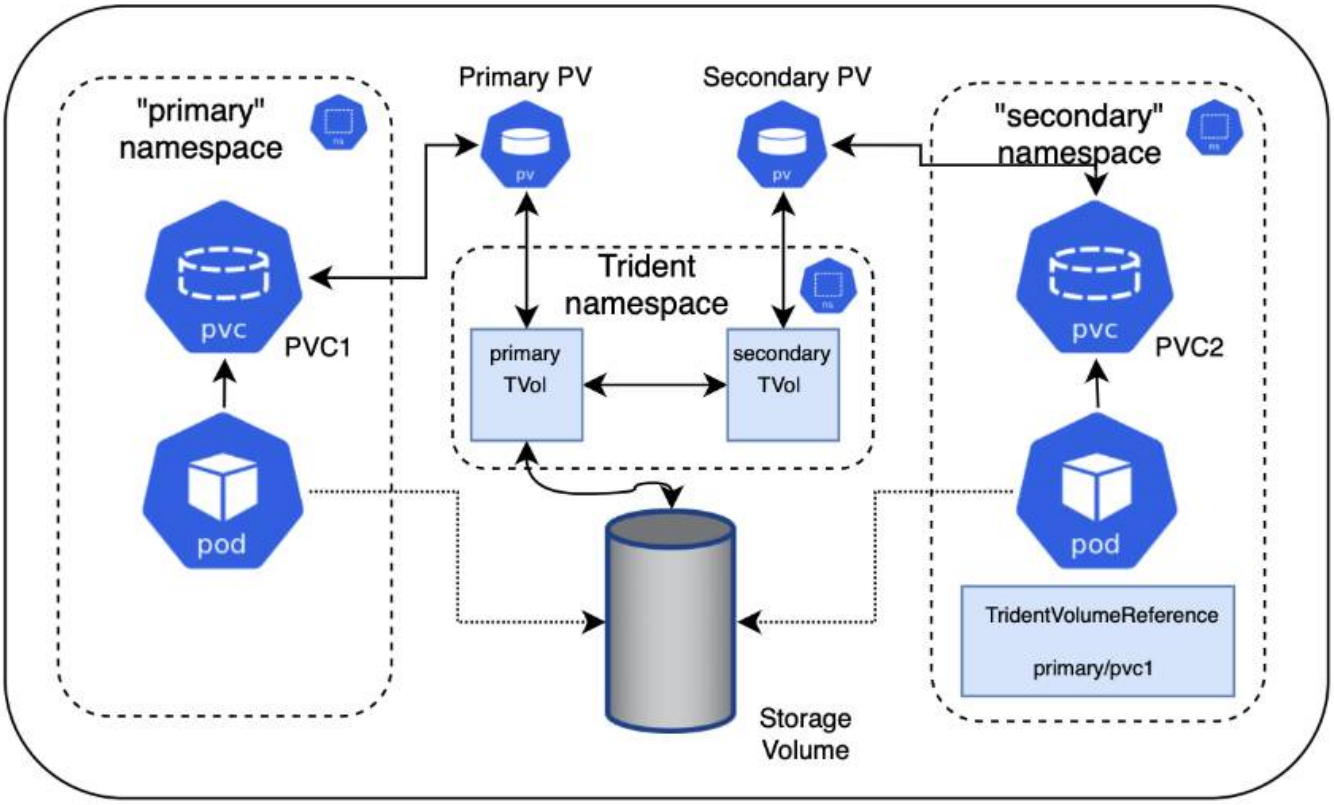
Com o Astra Trident, você pode criar um volume em um namespace principal e compartilhá-lo em um ou mais namespaces secundários.

Caraterísticas

O Astra TrigentVolumeReference CR permite que você compartilhe com segurança volumes NFS ReadWriteMany (RWX) em um ou mais namespaces do Kubernetes. Essa solução nativa do Kubernetes tem os seguintes benefícios:

- Vários níveis de controle de acesso para garantir a segurança
- Funciona com todos os drivers de volume Trident NFS
- Não há dependência do tridentctl ou de qualquer outro recurso do Kubernetes não nativo

Este diagrama ilustra o compartilhamento de volumes NFS em dois namespaces do Kubernetes.



Início rápido

Você pode configurar o compartilhamento de volume NFS em apenas algumas etapas.

1 **Configure o PVC de origem para compartilhar o volume**

O proprietário do namespace de origem concede permissão para acessar os dados no PVC de origem.

2 Conceda permissão para criar um CR no namespace de destino

O administrador do cluster concede permissão ao proprietário do namespace de destino para criar o CredentVolumeReference CR.

3 Crie TridentVolumeReference no namespace de destino

O proprietário do namespace de destino cria o TrigentVolumeReference CR para se referir ao PVC de origem.

4 Crie o PVC subordinado no namespace de destino

O proprietário do namespace de destino cria o PVC subordinado para usar a fonte de dados do PVC de origem.

Configure os namespaces de origem e destino

Para garantir a segurança, o compartilhamento entre namespace requer colaboração e ação do proprietário do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função de usuário é designada em cada etapa.

Passos

1. **Proprietário do namespace de origem:** Crie o PVC (`pvc1`) no namespace de origem que concede permissão para compartilhar com o namespace de destino (`namespace2`) usando a `shareToNamespace` anotação.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   O Astra Trident cria o PV e o volume de storage NFS no back-end.

- Você pode compartilhar o PVC para vários namespaces usando uma lista delimitada por vírgulas. Por exemplo, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
- Você pode compartilhar com todos os namespaces usando `*`. Por exemplo, `trident.netapp.io/shareToNamespace: *`
- Você pode atualizar o PVC para incluir a `shareToNamespace` anotação a qualquer momento.

2. **Cluster admin:** Crie a função personalizada e kubeconfig para conceder permissão ao proprietário do namespace de destino para criar o TridentVolumeReference CR no namespace de destino.
3. **Proprietário do namespace de destino:** Crie um CredentVolumeReference CR no namespace de destino que se refere ao namespace de origem `pvc1` .

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Proprietário do namespace de destino:** Crie um PVC (`pvc2`) no namespace de destino (`namespace2`) usando a `shareFromPVC` anotação para designar o PVC de origem.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

O tamanho do PVC de destino deve ser inferior ou igual ao PVC de origem.

Resultados

O Astra Trident lê a `shareFromPVC` anotação no PVC de destino e cria o PV de destino como um volume subordinado sem recurso de armazenamento próprio que aponta para o PV de origem e compartilha o recurso de armazenamento PV de origem. O PVC e o PV de destino aparecem encadernados normalmente.

Eliminar um volume partilhado

Você pode excluir um volume compartilhado entre vários namespaces. O Astra Trident removerá o acesso ao volume no namespace de origem e manterá acesso para outros namespaces que compartilham o volume. Quando todos os namespaces que fazem referência ao volume são removidos, o Astra Trident exclui o volume.

`tridentctl get`Use para consultar volumes subordinados

Usando o[`tridentctl` utilitário, você pode executar o `get` comando para obter volumes subordinados. Para obter mais informações, consulte o `tridentctl` comandos e opções.

```
Usage:
  tridentctl get [option]
```

Bandeiras -

- `-h, --help`: Ajuda para volumes.
- `--parentOfSubordinate string`: Limitar consulta ao volume de origem subordinado.
- `--subordinateOf string`: Limitar consulta a subordinados de volume.

Limitações

- O Astra Trident não pode impedir que namespaces de destino gravem no volume compartilhado. Você deve usar o bloqueio de arquivos ou outros processos para evitar a substituição de dados de volume compartilhado.
- Não é possível revogar o acesso ao PVC de origem removendo as `shareToNamespace` anotações ou ou `shareFromNamespace` excluindo o `TridentVolumeReference` CR. Para revogar o acesso, você deve excluir o PVC subordinado.
- Snapshots, clones e espelhamento não são possíveis em volumes subordinados.

Para mais informações

Para saber mais sobre o acesso ao volume entre namespace:

- "Compartilhamento de volumes entre namespaces: Diga olá ao acesso ao volume entre namespace"Visite .
- Assista à demonstração no "NetAppTV".

Use a topologia CSI

O Astra Trident pode criar e anexar volumes de forma seletiva a nós presentes em um cluster Kubernetes usando o "Recurso de topologia CSI".

Visão geral

Usando o recurso de topologia de CSI, o acesso a volumes pode ser limitado a um subconjunto de nós, com base em regiões e zonas de disponibilidade. Hoje em dia, os provedores de nuvem permitem que os administradores do Kubernetes gerem nós baseados em zonas. Os nós podem ser localizados em diferentes zonas de disponibilidade dentro de uma região ou em várias regiões. Para facilitar o provisionamento de volumes para workloads em uma arquitetura de várias zonas, o Astra Trident usa topologia de CSI.

Saiba mais sobre o recurso de topologia de CSI "aqui" .

O Kubernetes oferece dois modos exclusivos de vinculação de volume:

- `VolumeBindingMode`Com o definido como `Immediate`, o Astra Trident cria o volume sem qualquer reconhecimento de topologia. A vinculação de volume e o provisionamento dinâmico são tratados quando o PVC é criado. Esse é o padrão `VolumeBindingMode` e é adequado para clusters que não impõem restrições de topologia. Os volumes persistentes são criados sem depender dos requisitos de agendamento do pod solicitante.
- Com `VolumeBindingMode` definido como `WaitForFirstConsumer`, a criação e a vinculação de um volume persistente para um PVC é adiada até que um pod que usa o PVC seja programado e criado. Dessa forma, os volumes são criados para atender às restrições de agendamento impostas pelos requisitos de topologia.

O `WaitForFirstConsumer` modo de encadernação não requer rótulos de topologia. Isso pode ser usado independentemente do recurso de topologia de CSI.

O que você vai precisar

Para fazer uso da topologia de CSI, você precisa do seguinte:

- Um cluster de Kubernetes executando um "Versão do Kubernetes compatível"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Os nós no cluster devem ter rótulos que introduzam reconhecimento da topologia (`topology.kubernetes.io/region`e `topology.kubernetes.io/zone`). Esses rótulos **devem estar presentes nos nós no cluster** antes que o Astra Trident seja instalado para que o Astra Trident esteja ciente da topologia.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Etapa 1: Crie um back-end com reconhecimento de topologia

Os back-ends de storage do Astra Trident podem ser desenvolvidos para provisionar volumes de forma seletiva, com base nas zonas de disponibilidade. Cada back-end pode transportar um bloco opcional `supportedTopologies` que representa uma lista de zonas e regiões que devem ser suportadas. Para o StorageClasses que fazem uso de tal back-end, um volume só seria criado se solicitado por um aplicativo agendado em uma região/zona suportada.

Aqui está um exemplo de definição de backend:

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-a
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-b
```

JSON

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "backendName": "san-backend-us-east1",
 "managementLIF": "192.168.27.5",
 "svm": "iscsi_svm",
 "username": "admin",
 "password": "password",
 "supportedTopologies": [
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-a"},
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-b"}
]
}
```

`supportedTopologies` é usado para fornecer uma lista de regiões e zonas por backend. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em um StorageClass. Para os StorageClasses que contêm um subconjunto das regiões e zonas fornecidas em um back-end, o Astra Trident criará um volume no back-end.

Você também pode definir `supportedTopologies` por pool de armazenamento. Veja o exemplo a seguir:

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-a
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-b
storage:
- labels:
    workload: production
  region: Iowa-DC
  zone: Iowa-DC-A
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  region: Iowa-DC
  zone: Iowa-DC-B
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
```

Neste exemplo, as `region` etiquetas e `zone` representam a localização do conjunto de armazenamento. `topology.kubernetes.io/region` `topology.kubernetes.io/zone` e dite de onde os pools de storage podem ser consumidos.

Etapa 2: Defina StorageClasses que estejam cientes da topologia

Com base nas etiquetas de topologia fornecidas aos nós no cluster, o StorageClasses pode ser definido para conter informações de topologia. Isso determinará os pools de storage que atuam como candidatos a solicitações de PVC feitas e o subconjunto de nós que podem fazer uso dos volumes provisionados pelo Trident.

Veja o exemplo a seguir:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"
```

Na definição StorageClass fornecida acima, `volumeBindingMode` está definida como `WaitForFirstConsumer`. Os PVCs solicitados com este StorageClass não serão utilizados até que sejam referenciados em um pod. E, `allowedTopologies` fornece as zonas e a região a serem usadas. O `netapp-san-us-east1` StorageClass criará PVCs no `san-backend-us-east1` back-end definido acima.

Passo 3: Criar e usar um PVC

Com o StorageClass criado e mapeado para um back-end, agora você pode criar PVCs.

Veja o exemplo `spec` abaixo:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

Criar um PVC usando este manifesto resultaria no seguinte:

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Para o Trident criar um volume e vinculá-lo ao PVC, use o PVC em um pod. Veja o exemplo a seguir:

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Este podSpec instrui o Kubernetes a agendar o pod em nós presentes na `us-east1` região e escolher entre qualquer nó presente nas `us-east1-a` zonas ou `us-east1-b`.

Veja a seguinte saída:

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

Atualize os backends para incluir `supportedTopologies`

Os backends pré-existentes podem ser atualizados para incluir uma lista `supportedTopologies` de uso `tridentctl backend update`do . Isso não afetará os volumes que já foram provisionados e só será usado para PVCs subsequentes.

Encontre mais informações

- "Gerenciar recursos para contêineres"
- "NodeSeletor"
- "Afinidade e anti-afinidade"
- "Taints e Tolerations"

Trabalhar com instantâneos

Os snapshots de volume do Kubernetes de volumes persistentes (PVS) permitem cópias pontuais de volumes. Você pode criar um snapshot de um volume criado usando o Astra Trident, importar um snapshot criado fora do Astra Trident, criar um novo volume a partir de um snapshot existente e recuperar dados de volume de snapshots.

Visão geral

O instantâneo de volume é suportado por `ontap-nas` `ontap-nas-flexgroup` , , `ontap-san`, `ontap-san-economy`, , , , `solidfire-san` `gcp-cvs`, e `azure-netapp-files` drivers.

Antes de começar

Você deve ter um controlador de snapshot externo e definições personalizadas de recursos (CRDs) para trabalhar com snapshots. Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

Se a distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, Implantar um controlador de snapshot de volumeconsulte .

> Não crie um controlador de snapshot se estiver criando instantâneos de volume sob demanda em um ambiente GKE. O GKE usa um controlador instantâneo oculto integrado.

Criar um instantâneo de volume

Passos

1. Criar um `VolumeSnapshotClass`. para obter mais informações, "VolumeSnapshotClass"consulte .
   - O `driver` aponta para o condutor Astra Trident CSI.
   - `deletionPolicy` pode ser `Delete` ou `Retain`. Quando definido como `Retain`, o instantâneo físico subjacente no cluster de armazenamento é retido mesmo quando o `VolumeSnapshot` objeto é excluído.

     Exemplo

     ```
     cat snap-sc.yaml
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshotClass
     metadata:
       name: csi-snapclass
     driver: csi.trident.netapp.io
     deletionPolicy: Delete
     ```

2. Crie um instantâneo de um PVC existente.

   Exemplos

   - Este exemplo cria um instantâneo de um PVC existente.

     ```
     cat snap.yaml
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshot
     metadata:
       name: pvc1-snap
     spec:
       volumeSnapshotClassName: csi-snapclass
       source:
         persistentVolumeClaimName: pvc1
     ```

   - Este exemplo cria um objeto instantâneo de volume para um PVC chamado `pvc1` e o nome do instantâneo é definido como `pvc1-snap`. Um VolumeSnapshot é análogo a um PVC e está associado a um `VolumeSnapshotContent` objeto que representa o snapshot real.

     ```
     kubectl create -f snap.yaml
     volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

     kubectl get volumesnapshots
     NAME                    AGE
     pvc1-snap               50s
     ```

- Pode identificar o `VolumeSnapshotContent` objeto para o `pvc1-snap` VolumeSnapshot descrevendo-o. O `Snapshot Content Name` identifica o objeto VolumeSnapshotContent que serve este instantâneo. O `Ready To Use` parâmetro indica que o instantâneo pode ser usado para criar um novo PVC.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
.
.
```

Crie um PVC a partir de um instantâneo de volume

Você pode usar `dataSource` para criar um PVC usando um VolumeSnapshot nomeado `<pvc-name>` como a fonte dos dados. Depois que o PVC é criado, ele pode ser anexado a um pod e usado como qualquer outro PVC.

O PVC será criado no mesmo backend que o volume de origem. "KB: A criação de um PVC a partir de um instantâneo de PVC do Trident não pode ser criada em um back-end alternativo"Consulte a .

O exemplo a seguir cria o PVC usando `pvc1-snap` como fonte de dados.

```
cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Importar um instantâneo de volume

O Astra Trident é compatível com o "Processo de snapshot pré-provisionado do Kubernetes" para permitir que o administrador do cluster crie um `VolumeSnapshotContent` objeto e importe snapshots criados fora do Astra Trident.

Antes de começar

O Astra Trident precisa ter criado ou importado o volume pai do snapshot.

Passos

1. **Cluster admin:** Crie um `VolumeSnapshotContent` objeto que faça referência ao snapshot de back-end. Isso inicia o fluxo de trabalho de snapshot no Astra Trident.
   - Especifique o nome do instantâneo de back-end em `annotations` as `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Especifique `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` em `snapshotHandle`. esta é a única informação fornecida ao Astra Trident pelo snapshotter externo na `ListSnapshots` chamada.

     > O `<volumeSnapshotContentName>` nem sempre pode corresponder ao nome do instantâneo do back-end devido a restrições de nomenclatura CR.

     Exemplo

     O exemplo a seguir cria um `VolumeSnapshotContent` objeto que faz referência a snapshot de back-end `snap-01` .

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
```

2. **Cluster admin:** Crie o `VolumeSnapshot` CR que faz referência ao `VolumeSnapshotContent` objeto. Isso solicita acesso para usar o `VolumeSnapshot` em um namespace dado.

   Exemplo

   O exemplo a seguir cria um `VolumeSnapshot` CR chamado `import-snap` que faz referência ao `VolumeSnapshotContent` `import-snap-content` chamado .

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content
```

3. * Processamento interno (nenhuma ação necessária):* o Snapshotter externo reconhece o recém-criado `VolumeSnapshotContent` e executa a `ListSnapshots` chamada. O Astra Trident cria o `TridentSnapshot`.
   - O snapshotter externo define `VolumeSnapshotContent` para `readyToUse` e `VolumeSnapshot` para `true`.
   - Trident retorna `readyToUse=true`.
4. **Qualquer usuário:** Crie um `PersistentVolumeClaim` para fazer referência ao novo `VolumeSnapshot`, onde o `spec.dataSource` nome (ou `spec.dataSourceRef`) é o `VolumeSnapshot` nome.

   Exemplo

   O exemplo a seguir cria um PVC referenciando o `VolumeSnapshot` nome `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Recuperar dados de volume usando snapshots

O diretório instantâneo é oculto por padrão para facilitar a compatibilidade máxima dos volumes provisionados usando os `ontap-nas` drivers e `ontap-nas-economy`. Ative o `.snapshot` diretório para recuperar dados de instantâneos diretamente.

Use a CLI do ONTAP de restauração de snapshot de volume para restaurar um volume para um estado gravado em um snapshot anterior.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

Quando você restaura uma cópia snapshot, a configuração de volume existente é sobrescrita. As alterações feitas aos dados de volume após a criação da cópia instantânea são perdidas.

Eliminar um PV com instantâneos associados

Ao excluir um volume persistente com snapshots associados, o volume Trident correspondente é atualizado para um "estado de exclusão". Remova os snapshots de volume para excluir o volume Astra Trident.

Implantar um controlador de snapshot de volume

Se a sua distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, você poderá implantá-los da seguinte forma.

Passos

1. Criar CRDs de instantâneos de volume.

```
cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Crie o controlador instantâneo.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualize `namespace` para o seu namespace.

Links relacionados

- "Instantâneos de volume"
- "VolumeSnapshotClass"

Gerencie e monitore o Astra Trident

Atualizar o Astra Trident

Atualizar o Astra Trident

Começando com o lançamento de 24,02, o Astra Trident segue uma cadência de lançamento de quatro meses, entregando três grandes lançamentos todos os anos. Cada nova versão baseia-se nas versões anteriores e fornece novos recursos, melhorias de desempenho, correções de bugs e melhorias. Recomendamos que você atualize pelo menos uma vez por ano para aproveitar os novos recursos do Astra Trident.

Considerações antes da atualização

Ao atualizar para a versão mais recente do Astra Trident, considere o seguinte:

- Deve haver apenas uma instância do Astra Trident instalada em todos os namespaces em um determinado cluster do Kubernetes.
- O Astra Trident 23,07 e posterior requer snapshots de volume v1 e não é mais compatível com snapshots alfa ou beta.
- Se você criou o Cloud Volumes Service para Google Cloud no "Tipo de serviço CVS", atualize a configuração de back-end para usar o `standardsw` nível de serviço ou `zoneredundantstandardsw` ao atualizar a partir do Astra Trident 23,01. A falha ao atualizar o `serviceLevel` no back-end pode causar falha de volumes. "Amostras do tipo de serviço CVS"Consulte para obter detalhes.
- Ao atualizar, é importante que você forneça `parameter.fsType` o `StorageClasses` Astra Trident usado. Você pode excluir e recriar `StorageClasses` sem interromper volumes pré-existentes.
  - Este é um **requisito** para aplicação de "contextos de segurança" volumes SAN.
  - O diretório https://github.com/NetApp https://github.com/NetApp/Trident Trident/tree/master/Trident Trident-installer/sample-input[sample input] contém exemplos, como https://github.com/NetApp/Trident/blob/master/Trident-installer/sample-input/storage-class-samples/storage-class-[`storage-class-bronze-default.yaml`basic[`storage-class-basic.yaml.templ`]
  - Para obter mais informações, "Problemas conhecidos"consulte .

Passo 1: Selecione uma versão

As versões do Astra Trident seguem uma convenção de nomenclatura baseada em data `YY.MM`, onde "YY" é os últimos dois dígitos do ano e "MM" é o mês. Os lançamentos de ponto seguem uma `YY.MM.X` convenção, onde "X" é o nível de patch. Você selecionará a versão para a qual atualizar com base na versão da qual você está atualizando.

- Você pode fazer uma atualização direta para qualquer versão de destino que esteja dentro de uma janela de quatro versões da versão instalada. Por exemplo, você pode atualizar diretamente de 23,01 (ou qualquer lançamento de 23,01 pontos) para 24,02.
- Se você estiver atualizando de uma versão fora da janela de quatro versões, execute uma atualização em várias etapas. Use as instruções de atualização do "versão anterior"para atualizar para a versão mais recente que se encaixa na janela de quatro versões. Por exemplo, se você estiver executando o 22,01 e quiser atualizar para o 24,02:

a. Primeiro upgrade de 22,01 para 23,01.

b. Em seguida, atualize de 23,01 para 24,02.

> Ao atualizar usando o operador Trident na Plataforma de contêiner OpenShift, você deve atualizar para o Trident 21.01.1 ou posterior. O operador Trident lançado com 21.01.0 contém um problema conhecido que foi corrigido no 21.01.1. Para obter mais detalhes, consulte "Detalhes do problema no GitHub"a .

Passo 2: Determine o método de instalação original

Para determinar qual versão você usou para instalar originalmente o Astra Trident:

1. Use `kubectl get pods -n trident` para examinar os pods.
   - Se não houver nenhum pod do operador, o Astra Trident foi instalado usando `tridentctl`o .
   - Se houver um pod do operador, o Astra Trident foi instalado usando o operador Trident manualmente ou usando o Helm.
2. Se houver um pod do operador, use `kubectl describe torc` para determinar se o Astra Trident foi instalado usando o Helm.
   - Se houver uma etiqueta Helm, o Astra Trident foi instalado usando Helm.
   - Se não houver nenhuma etiqueta Helm, o Astra Trident foi instalado manualmente usando o operador Trident.

Passo 3: Selecione um método de atualização

Geralmente, você deve atualizar usando o mesmo método usado para a instalação inicial, no entanto, você pode "mova entre os métodos de instalação". Há duas opções para atualizar o Astra Trident.

- "Atualize usando o operador Trident"

> Sugerimos que você revise "Compreender o fluxo de trabalho de atualização do operador" antes de atualizar com o operador.

*

Atualize com o operador

Compreender o fluxo de trabalho de atualização do operador

Antes de usar o operador Trident para atualizar o Astra Trident, você deve entender os processos em segundo plano que ocorrem durante a atualização. Isso inclui alterações no controlador Trident, no pod de nó e no pod de nó e no DaemonSet que permitem atualizações contínuas.

Manuseio de atualização do operador Trident

Um dos muitos "Benefícios de usar o operador Trident" que instalar e atualizar o Astra Trident é o manuseio automático de objetos Kubernetes e Astra Trident sem interromper os volumes montados existentes. Dessa forma, o Astra Trident pode dar suporte a atualizações sem inatividade, ou "*atualizações contínuas*". Em particular, o operador do Trident se comunica com o cluster do Kubernetes para:

- Exclua e recrie a implantação do controlador Trident e o nó DaemonSet.
- Substitua o pod de nó Trident e o pod de nó Trident por novas versões.
  - Se um nó não for atualizado, ele não impedirá que os nós restantes sejam atualizados.
  - Somente os nós com um pod de nó Trident em execução podem montar volumes.

Para obter mais informações sobre a arquitetura Astra Trident no cluster Kubernetes, "A arquitetura do Astra Trident"consulte .

Fluxo de trabalho de atualização do operador

Quando você inicia uma atualização usando o operador Trident:

1. O operador **Trident**:
   a. Detecta a versão atualmente instalada do Astra Trident (versão *n*).
   b. Atualiza todos os objetos Kubernetes, incluindo CRDs, RBAC e Trident SVC.
   c. Exclui a implantação do controlador Trident para a versão *n*.
   d. Cria a implantação do controlador Trident para a versão *n-1*.
2. O Kubernetes* cria o pod de controlador Trident para *n-1*.
3. O operador **Trident**:
   a. Exclui o nó Trident DaemonSet para *n*. O operador não espera o encerramento do Node Pod.
   b. Cria o nó Trident Daemonset para *n-1*.
4. **Kubernetes** cria pods de nós do Trident em nós que não executam o Pod de nó do Trident *n*. Isso garante que nunca mais de um pod de nó Trident, de qualquer versão, em um nó.

Atualizar uma instalação do Astra Trident usando o operador Trident ou Helm

Você pode atualizar o Astra Trident usando o operador Trident manualmente ou usando o Helm. Você pode atualizar de uma instalação de operador Trident para outra instalação de operador Trident ou atualizar de uma `tridentctl` instalação para uma versão de operador Trident. Reveja "Selecione um método de atualização" antes de atualizar a instalação de um operador Trident.

Atualize uma instalação manual

Você pode atualizar de uma instalação de operador Trident com escopo de cluster para outra instalação de operador Trident com escopo de cluster. Todos os Astra Trident versões 21,01 e superiores usam um operador com escopo de cluster.

Para atualizar do Astra Trident que foi instalado usando o operador com escopo de namespace (versões 20,07 a 20,10), use as instruções de atualização do "sua versão instalada"Astra Trident.

Sobre esta tarefa

O Trident fornece um arquivo de pacote que você pode usar para instalar o operador e criar objetos associados para sua versão do Kubernetes.

- Para clusters que executam o Kubernetes 1,24 ou anterior, "bundle_pre_1_25.yaml"use o .
- Para clusters que executam o Kubernetes 1,25 ou posterior, "bundle_post_1_25.yaml"use o .

Antes de começar

Verifique se você está usando um cluster do Kubernetes executando "Uma versão compatível do Kubernetes"o .

Passos

1. Verifique sua versão do Astra Trident:

```
./tridentctl -n trident version
```

2. Exclua o operador Trident que foi usado para instalar a instância atual do Astra Trident. Por exemplo, se você estiver atualizando do 23,07, execute o seguinte comando:

```
kubectl delete -f 23.07.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

3. Se você personalizou sua instalação inicial usando `TridentOrchestrator` atributos, você pode editar o `TridentOrchestrator` objeto para modificar os parâmetros de instalação. Isso pode incluir alterações feitas para especificar Registros de imagens Trident e CSI espelhados para o modo offline, habilitar logs de depuração ou especificar segredos de recebimento de imagens.
4. Instale o Astra Trident usando o arquivo YAML do pacote correto para o seu ambiente, onde *<bundle.yaml>* é `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` baseado na sua versão do Kubernetes. Por exemplo, se você estiver instalando o Astra Trident 24,02, execute o seguinte comando:

```
kubectl create -f 24.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

Atualize uma instalação do Helm

Você pode atualizar uma instalação do Astra Trident Helm.

> Ao atualizar um cluster do Kubernetes do 1,24 para o 1,25 ou posterior que tenha o Astra Trident instalado, você deve atualizar o Values.yaml para definir `excludePodSecurityPolicy` `true` ou adicionar `--set excludePodSecurityPolicy=true` `helm upgrade` ao comando antes de atualizar o cluster.

Passos

1. Se "Instalado Astra Trident usando o Helm"você , você pode usar `helm upgrade trident netapp-trident/trident-operator --version 100.2402.0` o para atualizar em uma etapa. Se você não adicionou o repositório Helm ou não pode usá-lo para atualizar:
   a. Baixe o mais recente lançamento do Astra Trident em "A seção *assets* no GitHub".

b. Use o `helm upgrade` comando onde `trident-operator-24.02.0.tgz` reflete a versão para a qual você deseja atualizar.

```
helm upgrade <name> trident-operator-24.02.0.tgz
```

Se você definir opções personalizadas durante a instalação inicial (como especificar Registros privados e espelhados para imagens Trident e CSI), anexe o `helm upgrade` comando usando `--set` para garantir que essas opções estejam incluídas no comando upgrade, caso contrário, os valores serão redefinidos para padrão.

2. Execute `helm list` para verificar se o gráfico e a versão do aplicativo foram atualizados. Execute `tridentctl logs` para rever todas as mensagens de depuração.

Atualize de uma `tridentctl` instalação para o operador Trident

Pode atualizar para a versão mais recente do operador Trident a partir de uma `tridentctl` instalação. Os backends e PVCs existentes estarão automaticamente disponíveis.

Antes de alternar entre os métodos de instalação, revise "Movendo-se entre os métodos de instalação"o .

Passos

1. Baixe o mais recente lançamento do Astra Trident.

```
# Download the release required [24.020.0]
mkdir 24.02.0
cd 24.02.0
wget
https://github.com/NetApp/trident/releases/download/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.tar.gz
cd trident-installer
```

2. Crie o `tridentorchestrator` CRD a partir do manifesto.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Implante o operador com escopo de cluster no mesmo namespace.

```
kubectl create -f deploy/<bundle-name.yaml>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. Crie `TridentOrchestrator` um CR para a instalação do Astra Trident.

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Confirme se o Trident foi atualizado para a versão pretendida.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v24.02.0
```

Atualize com o tridentctl

É possível atualizar facilmente uma instalação existente do Astra Trident usando `tridentctl`o .

Sobre esta tarefa

Desinstalar e reinstalar o Astra Trident funciona como uma atualização. Quando você desinstalar o Trident, a reivindicação de volume persistente (PVC) e o volume persistente (PV) usados pela implantação do Astra Trident não são excluídos. Os PVS que já tiverem sido provisionados permanecerão disponíveis enquanto o Astra Trident estiver offline, e o Astra Trident provisionará volumes para quaisquer PVCs que forem criados nesse período, uma vez que estiverem novamente online.

Antes de começar

Revise "Selecione um método de atualização" antes de atualizar usando `tridentctl`o .

Passos

1. Execute o comando uninstall `tridentctl` para remover todos os recursos associados ao Astra Trident, exceto para CRDs e objetos relacionados.

```
./tridentctl uninstall -n <namespace>
```

2. Reinstale o Astra Trident. "Instale o Astra Trident usando o tridentctl"Consulte a .

> Não interrompa o processo de atualização. Certifique-se de que o instalador é executado até a conclusão.

Gerenciar o Astra Trident usando o tridentctl

O "Pacote de instalação do Trident" inclui o `tridentctl` utilitário de linha de comando para fornecer acesso simples ao Astra Trident. Usuários do Kubernetes com Privileges suficiente podem usá-lo para instalar o Astra Trident ou gerenciar o namespace que contém o pod Astra Trident.

Comandos e sinalizadores globais

Você pode executar `tridentctl help` para obter uma lista de comandos disponíveis `tridentctl` ou anexar o `--help` sinalizador a qualquer comando para obter uma lista de opções e sinalizadores para esse comando específico.

```
tridentctl [command] [--optional-flag]
```

O utilitário Astra Trident `tridentctl` suporta os seguintes comandos e sinalizadores globais.

Comandos

`create`
Adicionar um recurso ao Astra Trident.

`delete`
Remova um ou mais recursos do Astra Trident.

`get`
Obtenha um ou mais recursos do Astra Trident.

`help`
Ajuda sobre qualquer comando.

`images`
Imprima uma tabela das imagens de contêiner que o Astra Trident precisa.

`import`
Importar um recurso existente para o Astra Trident.

`install`
Instale o Astra Trident.

`logs`
Imprima os logs do Astra Trident.

`send`
Enviar um recurso do Astra Trident.

`uninstall`
Desinstale o Astra Trident.

`update`
Modificar um recurso no Astra Trident.

`update backend state`
Suspender temporariamente as operações de back-end.

`upgrade`
Atualizar um recurso no Astra Trident.

`version`
Imprima a versão do Astra Trident.

Bandeiras globais

`-d, --debug`
Saída de depuração.

`-h, --help`
Ajuda para `tridentctl`.

`-k, --kubeconfig string`
Especifique `KUBECONFIG` o caminho para executar comandos localmente ou de um cluster do Kubernetes para outro.

Como alternativa, você pode exportar a `KUBECONFIG` variável para apontar para um cluster Kubernetes específico e emitir `tridentctl` comandos para esse cluster.

`-n, --namespace string`
Namespace da implantação do Astra Trident.

`-o, --output string`
Formato de saída. Um de JSON|yaml|name|wide|ps (padrão).

`-s, --server string`
Endereço/porta da interface REST do Astra Trident.

A interface REST DO Trident pode ser configurada para ouvir e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

Opções de comando e sinalizadores

criar

Use o `create` comando para adicionar um recurso ao Astra Trident.

```
tridentctl create [option]
```

Opções

`backend`: Adicionar um back-end ao Astra Trident.

eliminar

Use o `delete` comando para remover um ou mais recursos do Astra Trident.

```
tridentctl delete [option]
```

Opções

`backend`: Excluir um ou mais back-ends de storage do Astra Trident.
`snapshot`: Excluir um ou mais snapshots de volume do Astra Trident.
`storageclass`: Excluir uma ou mais classes de storage do Astra Trident.

`volume`: Excluir um ou mais volumes de storage do Astra Trident.

obter

Use o `get` comando para obter um ou mais recursos do Astra Trident.

```
tridentctl get [option]
```

Opções

`backend`: Obtenha um ou mais back-ends de storage do Astra Trident.
`snapshot`: Obtenha um ou mais snapshots do Astra Trident.
`storageclass`: Obtenha uma ou mais classes de storage do Astra Trident.
`volume`: Obtenha um ou mais volumes do Astra Trident.

Bandeiras

`-h`, `--help`: Ajuda para volumes.
`--parentOfSubordinate string`: Limitar consulta ao volume de origem subordinado.
`--subordinateOf string`: Limitar consulta a subordinados de volume.

imagens

Use `images` sinalizadores para imprimir uma tabela das imagens de contêiner que o Astra Trident precisa.

```
tridentctl images [flags]
```

Bandeiras

`-h`, `--help`: Ajuda para imagens.
`-v --k8s-version string`, : Versão semântica do cluster do Kubernetes.

importar volume

Use o `import volume` comando para importar um volume existente para o Astra Trident.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

Aliases

`volume`, `v`

Bandeiras

`-f --filename string`, : Caminho para o arquivo PVC YAML ou JSON.
`-h`, `--help`: Ajuda para volume.
`--no-manage`: Criar apenas PV/PVC. Não assuma o gerenciamento do ciclo de vida do volume.

instale

Use os `install` sinalizadores para instalar o Astra Trident.

```
tridentctl install [flags]
```

Bandeiras

`--autosupport-image string`: A imagem do contentor para telemetria AutoSupport (predefinição "NetApp/Trident AutoSupport:<current-version>").
`--autosupport-proxy string`: O endereço/porta de um proxy para o envio de telemetria AutoSupport.
`--enable-node-prep`: Tentativa de instalar os pacotes necessários nos nós.
`--generate-custom-yaml`: Gere arquivos YAML sem instalar nada.
`-h --help`, : Ajuda para instalar.
`--http-request-timeout`: Substituir o tempo limite da solicitação HTTP para a API REST do controlador Trident (1m30s padrão).
`--image-registry string`: O endereço/porta de um Registro de imagem interno.
`--k8s-timeout duration`: O tempo limite para todas as operações do Kubernetes (3m0s padrão).
`--kubelet-dir string`: A localização do host do estado interno do kubelet (padrão "/var/lib/kubelet").
`--log-format string`: O formato de log Astra Trident (texto, json) (texto padrão).
`--pv string`: O nome do PV legado usado pelo Astra Trident garante que isso não existe (padrão "Trident").
`--pvc string`: O nome do PVC legado usado pelo Astra Trident garante que isso não existe (padrão "Trident").
`--silence-autosupport`: Não envie pacotes AutoSupport automaticamente para o NetApp (padrão verdadeiro).
`--silent`: Desativar a saída MOST durante a instalação.
`--trident-image string`: A imagem Astra Trident a instalar.
`--use-custom-yaml`: Use todos os arquivos YAML existentes que existem no diretório de configuração.
`--use-ipv6`: Utilizar o IPv6 para a comunicação do Astra Trident.

registos

Use `logs` sinalizadores para imprimir os logs do Astra Trident.

```
tridentctl logs [flags]
```

Bandeiras

`-a, --archive`: Crie um arquivo de suporte com todos os logs, a menos que especificado de outra forma.
`-h --help`, : Ajuda para logs.
`-l --log string`, : Log do Astra Trident para exibição. Um dos Trident|auto|Trident-operator|All (predefinição "auto").
`--node string`: O nome do nó Kubernetes do qual você pode coletar logs do pod de nó.
`-p --previous`, : Obtém os registos para a instância de contentor anterior, se existir.
`--sidecars`: Obter os logs para os recipientes sidecar.

enviar

Use o `send` comando para enviar um recurso do Astra Trident.

```
tridentctl send [option]
```

Opções

`autosupport`: Enviar um arquivo AutoSupport para o NetApp.

desinstalar

Use `uninstall` sinalizadores para desinstalar o Astra Trident.

```
tridentctl uninstall [flags]
```

Bandeiras

`-h, --help`: Ajuda para desinstalar.
`--silent`: Desativar a saída MOST durante a desinstalação.

atualização

Use o `update` comando para modificar um recurso no Astra Trident.

```
tridentctl update [option]
```

Opções

`backend`: Atualize um back-end no Astra Trident.

atualizar estado de back-end

Use o `update backend state` comando para suspender ou retomar as operações de back-end.

```
tridentctl update backend state <backend-name> [flag]
```

Bandeiras

`-h --help`, : Ajuda para o estado de back-end.
`--user-state`: Defina como `suspended` para pausar operações de back-end. Defina como `normal` para retomar as operações de back-end. Quando definido para `suspended`:

- `AddVolume` `CloneVolume`, , `Import Volume`, `ResizeVolume` estão em pausa.
- `PublishVolume` `UnPublishVolume`, , `CreateSnapshot` `GetSnapshot` , `RestoreSnapshot`, , , `DeleteSnapshot` `RemoveVolume`, , `GetVolumeExternal`, `ReconcileNodeAccess` permanecem disponíveis.

versão

Use `version` sinalizadores para imprimir a versão do `tridentctl` e o serviço Trident em execução.

```
tridentctl version [flags]
```

Bandeiras

`--client`: Somente versão do cliente (nenhum servidor necessário).
`-h, --help`: Ajuda para a versão.

Monitore o Astra Trident

O Astra Trident fornece um conjunto de pontos de extremidade de métricas Prometheus que você pode usar para monitorar a performance do Astra Trident.

Visão geral

As métricas fornecidas pelo Astra Trident permitem que você faça o seguinte:

- Acompanhe a integridade e a configuração do Astra Trident. Você pode examinar como as operações são bem-sucedidas e se elas podem se comunicar com os backends como esperado.
- Examine as informações de uso do back-end e entenda quantos volumes são provisionados em um back-end e a quantidade de espaço consumido, etc.
- Mantenha um mapeamento da quantidade de volumes provisionados em backends disponíveis.
- Acompanhe o desempenho. Você pode ver quanto tempo leva para que o Astra Trident se comunique com back-ends e realize operações.

> Por padrão, as métricas do Trident são expostas na porta de destino `8001` no `/metrics` endpoint. Essas métricas são **ativadas por padrão** quando o Trident está instalado.

O que você vai precisar

- Um cluster Kubernetes com Astra Trident instalado.
- Uma instância Prometheus. Isso pode ser um "Implantação do Prometheus em contêiner" ou você pode optar por executar Prometheus como um "aplicação nativa".

Passo 1: Defina um alvo Prometheus

Você deve definir um alvo Prometheus para reunir as métricas e obter informações sobre os back-ends que o Astra Trident gerencia, os volumes que ele cria e assim por diante. "blog"Isso explica como você pode usar Prometheus e Grafana com o Astra Trident para recuperar métricas. O blog explica como você pode executar o Prometheus como um operador no cluster Kubernetes e a criação de um ServiceMonitor para obter métricas do Astra Trident.

Passo 2: Crie um Prometheus ServiceMonitor

Para consumir as métricas do Trident, você deve criar um Prometheus ServiceMonitor que vigia `trident-csi` o serviço e escuta na `metrics` porta. Um exemplo de ServiceMonitor se parece com isso:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
      release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
    - trident
  endpoints:
  - port: metrics
    interval: 15s
```

Essa definição do ServiceMonitor recupera as métricas retornadas pelo `trident-csi` serviço e procura especificamente o `metrics` ponto final do serviço. Como resultado, prometeu agora está configurado para entender as métricas do Astra Trident.

Além das métricas disponíveis diretamente do Astra Trident, o kubelet expõe muitas `kubelet_volume_*` métricas por meio do seu próprio ponto de extremidade de métricas. O Kubelet pode fornecer informações sobre os volumes anexados e pods e outras operações internas que ele manipula. Consulte a "aqui".

Passo 3: Consultar métricas do Trident com PromQL

PromQL é bom para criar expressões que retornam dados de séries temporais ou tabulares.

Aqui estão algumas consultas PromQL que você pode usar:

Obtenha informações de saúde do Trident

- **Porcentagem de respostas HTTP 2XX do Astra Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Porcentagem de RESPOSTAS REST do Astra Trident via código de status**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Duração média em ms das operações realizadas pelo Astra Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Obtenha informações de uso do Astra Trident

- **Tamanho médio do volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Espaço total de volume provisionado por cada back-end**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Obtenha uso de volume individual

Isso é ativado somente se as métricas do kubelet também forem coletadas.

- **Porcentagem de espaço usado para cada volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Saiba mais sobre a telemetria do Astra Trident AutoSupport

Por padrão, o Astra Trident envia métricas de Prometheus e informações básicas de back-end para o NetApp em uma cadência diária.

- Para impedir que o Astra Trident envie métricas e informações básicas de back-end para o NetApp, passe a `--silence-autosupport` bandeira durante a instalação do Astra Trident.
- O Astra Trident também pode enviar logs de contêiner para o suporte do NetApp sob demanda por meio `tridentctl send autosupport`do . Você precisará acionar o Astra Trident para fazer o upload dos seus logs. Antes de enviar logs, você deve aceitar o NetApp'"política de privacidade"s .
- A menos que especificado, o Astra Trident obtém os logs das últimas 24 horas.
- Você pode especificar o período de tempo de retenção do log com o `--since` sinalizador. Por exemplo `tridentctl send autosupport --since=1h`: . Essas informações são coletadas e enviadas por meio `trident-autosupport` de um contêiner que é instalado ao lado do Astra Trident. Pode obter a imagem do contentor em "Trident AutoSupport".
- A Trident AutoSupport não coleta nem transmite informações de identificação pessoal (PII) ou informações pessoais. Ele vem com um "EULA" que não é aplicável à própria imagem de contentor Trident. Você pode saber mais sobre o compromisso da NetApp com a segurança e a confiança dos dados "aqui" .

Um exemplo de payload enviado pelo Astra Trident é parecido com este:

```
---
items:
- backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
  protocol: file
  config:
    version: 1
    storageDriverName: ontap-nas
    debug: false
    debugTraceFlags:
    disableDelete: false
    serialNumbers:
    - nwkvzfanek_SN
    limitVolumeSize: ''
  state: online
  online: true
```

- As mensagens do AutoSupport são enviadas para o ponto de extremidade do AutoSupport do NetApp. Se você estiver usando um Registro privado para armazenar imagens de contentor, você pode usar o `--image-registry` sinalizador.
- Você também pode configurar URLs de proxy gerando os arquivos YAML de instalação. Isso pode ser feito usando `tridentctl install --generate-custom-yaml` para criar os arquivos YAML e adicionar o `--proxy-url` argumento para o `trident-autosupport` contentor no `trident-deployment.yaml`.

Desativar métricas do Astra Trident

Para **desabilitar métricas** de serem reportadas, você deve gerar YAMLs personalizados (usando o `--generate-custom-yaml` sinalizador) e editá-los para remover o `--metrics` sinalizador de ser invocado para o `trident-main` contentor.

Desinstale o Astra Trident

Você deve usar o mesmo método para desinstalar o Astra Trident usado para instalar o Astra Trident.

Sobre esta tarefa

- Se você precisar de uma correção para bugs observados após uma atualização, problemas de dependência ou uma atualização mal sucedida ou incompleta, você deve desinstalar o Astra Trident e reinstalar a versão anterior usando as instruções específicas para isso "versão". Esta é a única maneira recomendada de *downgrade* para uma versão anterior.
- Para facilitar a atualização e reinstalação, a desinstalação do Astra Trident não remove os CRDs ou objetos relacionados criados pelo Astra Trident. Se você precisar remover completamente o Astra Trident e todos os seus dados, "Remova completamente o Astra Trident e CRDs"consulte .

Antes de começar

Se você está desativando clusters do Kubernetes, exclua todas as aplicações que usam volumes criados pelo

Astra Trident antes da desinstalação. Isso garante que os PVCs sejam inéditos nos nós do Kubernetes antes que sejam excluídos.

Determine o método de instalação original

Você deve usar o mesmo método para desinstalar o Astra Trident que você usou para instalá-lo. Antes de desinstalar, verifique qual versão você usou para instalar originalmente o Astra Trident.

1. Use `kubectl get pods -n trident` para examinar os pods.
   - Se não houver nenhum pod do operador, o Astra Trident foi instalado usando `tridentctl`o .
   - Se houver um pod do operador, o Astra Trident foi instalado usando o operador Trident manualmente ou usando o Helm.
2. Se houver um pod do operador, use `kubectl describe tproc trident` para determinar se o Astra Trident foi instalado usando o Helm.
   - Se houver uma etiqueta Helm, o Astra Trident foi instalado usando Helm.
   - Se não houver nenhuma etiqueta Helm, o Astra Trident foi instalado manualmente usando o operador Trident.

Desinstale a instalação de um operador Trident

Você pode desinstalar manualmente uma instalação do operador do Trident ou usando o Helm.

Desinstalar a instalação manual

Se você instalou o Astra Trident usando o operador, você pode desinstalá-lo fazendo um dos seguintes procedimentos:

1. **Editar `TridentOrchestrator` CR e definir o sinalizador de desinstalação**:

   ```
   kubectl patch torc <trident-orchestrator-name> --type=merge -p
   '{"spec":{"uninstall":true}}'
   ```

   Quando o `uninstall` sinalizador está definido como `true`, o operador Trident desinstala o Trident, mas não remove o próprio TridentOrchestrator. Você deve limpar o TridentOrchestrator e criar um novo se quiser instalar o Trident novamente.

2. **Excluir `TridentOrchestrator`**: Ao remover o `TridentOrchestrator` CR que foi usado para implantar o Astra Trident, você instrui o operador a desinstalar o Trident. O operador processa a remoção `TridentOrchestrator` e remove a implantação do Astra Trident e o daemonset, excluindo os pods do Trident que ele criou como parte da instalação.

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

Desinstale a instalação do Helm

Se você instalou o Astra Trident usando o Helm, você pode desinstalá-lo usando `helm uninstall`o .

```
#List the Helm release corresponding to the Astra Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

Desinstale uma `tridentctl` instalação

Use o `uninstall` comando in `tridentctl` para remover todos os recursos associados ao Astra Trident, exceto para CRDs e objetos relacionados:

```
./tridentctl uninstall -n <namespace>
```

Astra Trident para Docker

Pré-requisitos para implantação

Você precisa instalar e configurar os pré-requisitos de protocolo necessários no seu host antes de implantar o Astra Trident.

Verifique os requisitos

- Verifique se sua implantação atende a todos "requisitos"os .
- Verifique se você tem uma versão suportada do Docker instalada. Se a versão do Docker estiver desatualizada, "instale ou atualize-o.".

```
docker --version
```

- Verifique se os pré-requisitos do protocolo estão instalados e configurados no seu host.

Ferramentas NFS

Instale as ferramentas NFS usando os comandos do seu sistema operacional.

RHEL 8 MAIS

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

Reinicie seus nós de trabalho após instalar as ferramentas NFS para evitar falhas ao anexar volumes a contêineres.

Ferramentas iSCSI

Instale as ferramentas iSCSI utilizando os comandos do seu sistema operativo.

RHEL 8 MAIS

1. Instale os seguintes pacotes de sistema:

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-
mapper-multipath
```

2. Verifique se a versão iscsi-iniciador-utils é 6,2.0,874-2.el7 ou posterior:

```
rpm -q iscsi-initiator-utils
```

3. Definir a digitalização para manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ativar multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de `etc/multipath.conf` que contém `find_multipaths no defaults` em .

5. Certifique-se de que `iscsid` e `multipathd` estão a funcionar:

```
sudo systemctl enable --now iscsid multipathd
```

6. Ativar e iniciar `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Instale os seguintes pacotes de sistema:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Verifique se a versão Open-iscsi é 2,0.874-5ubuntu2.10 ou posterior (para bionic) ou 2,0.874-7.1ubuntu6.1 ou posterior (para focal):

```
dpkg -l open-iscsi
```

3. Definir a digitalização para manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ativar multipathing:

```
sudo tee /etc/multipath.conf <<-'EOF'
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de `etc/multipath.conf` que contém `find_multipaths no` `defaults` em .

5. Certifique-se de que `open-iscsi` e `multipath-tools` estão ativados e em execução:

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

Ferramentas NVMe

Instale as ferramentas NVMe usando os comandos do seu sistema operacional.

- O NVMe requer o RHEL 9 ou posterior.
- Se a versão do kernel do seu nó Kubernetes for muito antiga ou se o pacote NVMe não estiver disponível para a versão do kernel, talvez seja necessário atualizar a versão do kernel do nó para uma com o pacote NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Implante o Astra Trident

O Astra Trident para Docker fornece integração direta com o ecossistema Docker para plataformas de storage NetApp. Ele dá suporte ao provisionamento e gerenciamento de recursos de storage da plataforma de storage para hosts Docker, com uma estrutura para adicionar plataformas adicionais no futuro.

Várias instâncias do Astra Trident podem ser executadas simultaneamente no mesmo host. Isso permite conexões simultâneas a vários sistemas de armazenamento e tipos de armazenamento, com a capacidade de personalizar o armazenamento usado para os volumes Docker.

O que você vai precisar

Consulte "pré-requisitos para implantação". Depois de garantir que os pré-requisitos sejam atendidos, você estará pronto para implantar o Astra Trident.

Método de plug-in gerenciado Docker (versão 1,13/17,03 e posterior)

Antes de começar

Se você usou o Astra Trident pré Docker 1,13/17,03 no método daemon tradicional, certifique-se de parar o processo Astra Trident e reiniciar seu daemon Docker antes de usar o método do plugin gerenciado.

1. Parar todas as instâncias em execução:

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Reinicie o Docker.

```
systemctl restart docker
```

3. Certifique-se de que tem o Docker Engine 17,03 (novo 1,13) ou posterior instalado.

```
docker --version
```

Se a sua versão estiver desatualizada, "instale ou atualize a instalação".

Passos

1. Crie um arquivo de configuração e especifique as opções da seguinte forma:
   - `config`: O nome do arquivo padrão é `config.json`, no entanto, você pode usar qualquer nome que você escolher especificando a `config` opção com o nome do arquivo. O arquivo de configuração deve estar localizado `/etc/netappdvp` no diretório no sistema host.
   - `log-level`: Especifique o nível de registo (`debug`, `info`, `warn` `error` , , `fatal`). A predefinição é `info`.
   - `debug`: Especifique se o log de depuração está ativado. O padrão é falso. Substitui o nível de log, se verdadeiro.
     i. Crie um local para o arquivo de configuração:

     ```
     sudo mkdir -p /etc/netappdvp
     ```

     ii. Crie o arquivo de configuração:

     ```
     cat << EOF > /etc/netappdvp/config.json
     {
         "version": 1,
         "storageDriverName": "ontap-nas",
         "managementLIF": "10.0.0.1",
         "dataLIF": "10.0.0.2",
         "svm": "svm_nfs",
         "username": "vsadmin",
         "password": "password",
         "aggregate": "aggr1"
     }
     EOF
     ```

2. Inicie o Astra Trident usando o sistema de plugins gerenciado. Substitua `<version>` pela versão do plugin (xxx.xx.x) que você está usando.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. Comece a usar o Astra Trident para consumir storage do sistema configurado.

a. Crie um volume chamado "firstvolume":

```
docker volume create -d netapp --name firstVolume
```

b. Crie um volume padrão quando o contentor for iniciado:

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

c. Remover o volume "firstvolume":

```
docker volume rm firstVolume
```

Método tradicional (versão 1,12 ou anterior)

Antes de começar

1. Certifique-se de que você tem o Docker versão 1,10 ou posterior.

```
docker --version
```

Se a sua versão estiver desatualizada, atualize a instalação.

```
curl -fsSL https://get.docker.com/ | sh
```

Ou, "siga as instruções para sua distribuição".

2. Certifique-se de que NFS e/ou iSCSI estão configurados para o seu sistema.

Passos

1. Instale e configure o plug-in de volume do Docker do NetApp:
   a. Baixe e descompacte o aplicativo:

```
wget
https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.02.0.tar.gz
tar zxf trident-installer-24.02.0.tar.gz
```

   b. Mover para um local no caminho do compartimento:

```
sudo mv trident-installer/extras/bin/trident /usr/local/bin/
sudo chown root:root /usr/local/bin/trident
sudo chmod 755 /usr/local/bin/trident
```

c. Crie um local para o arquivo de configuração:

```
sudo mkdir -p /etc/netappdvp
```

d. Crie o arquivo de configuração:

```
cat << EOF > /etc/netappdvp/ontap-nas.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1"
}
EOF
```

2. Depois de colocar o binário e criar o arquivo de configuração, inicie o daemon Trident usando o arquivo de configuração desejado.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

A menos que especificado, o nome padrão para o driver de volume é "NetApp".

Depois que o daemon é iniciado, você pode criar e gerenciar volumes usando a interface CLI do Docker

3. Criar um volume:

```
docker volume create -d netapp --name trident_1
```

4. Provisione um volume Docker ao iniciar um contentor:

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Remover um volume Docker:

```
docker volume rm trident_1
docker volume rm trident_2
```

Inicie o Astra Trident na inicialização do sistema

Um arquivo de unidade de exemplo para sistemas baseados em systemd pode ser encontrado `contrib/trident.service.example` no repositório Git. Para usar o arquivo com RHEL, faça o seguinte:

1. Copie o arquivo para o local correto.

   Você deve usar nomes exclusivos para os arquivos de unidade se tiver mais de uma instância em execução.

```
cp contrib/trident.service.example
/usr/lib/systemd/system/trident.service
```

2. Edite o arquivo, altere a descrição (linha 2) para corresponder ao nome do driver e ao caminho do arquivo de configuração (linha 9) para refletir seu ambiente.
3. Recarregue systemd para que ele ingere alterações:

```
systemctl daemon-reload
```

4. Ative o serviço.

   Esse nome varia dependendo do que você nomeou o arquivo no `/usr/lib/systemd/system` diretório.

```
systemctl enable trident
```

5. Inicie o serviço.

```
systemctl start trident
```

6. Ver o estado.

```
systemctl status trident
```

Sempre que você modificar o arquivo unit, execute o `systemctl daemon-reload` comando para que ele esteja ciente das alterações.

Atualize ou desinstale o Astra Trident

Você pode atualizar com segurança o Astra Trident para Docker sem qualquer impacto nos volumes que estão em uso. Durante o processo de atualização, haverá um breve período em que `docker volume` os comandos direcionados para o plugin não serão bem-sucedidos, e os aplicativos não poderão montar volumes até que o plugin esteja sendo executado novamente. Na maioria das circunstâncias, esta é uma questão de segundos.

Atualização

Execute as etapas abaixo para atualizar o Astra Trident para Docker.

Passos

1. Listar os volumes existentes:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. Desativar o plugin:

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Atualize o plugin:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> O lançamento de 18,01 do Astra Trident substitui o nDVP. Você deve atualizar diretamente da `netapp/ndvp-plugin` imagem para a `netapp/trident-plugin` imagem.

4. Ativar o plugin:

```
docker plugin enable netapp:latest
```

5. Verifique se o plugin está ativado:

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. Verifique se os volumes estão visíveis:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

> Se você estiver atualizando de uma versão antiga do Astra Trident (pré-20,10) para Astra Trident 20,10 ou posterior, talvez haja um erro. Para obter mais informações, "Problemas conhecidos"consulte . Se você correr para o erro, você deve primeiro desativar o plugin, em seguida, remover o plugin e, em seguida, instalar a versão necessária do Astra Trident passando um parâmetro de configuração extra: `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Desinstalar

Execute as etapas abaixo para desinstalar o Astra Trident para Docker.

Passos

1. Remova todos os volumes criados pelo plugin.
2. Desativar o plugin:

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Remova o plugin:

```
docker plugin rm netapp:latest
```

Trabalhe com volumes

Você pode criar, clonar e remover volumes facilmente usando os comandos padrão

`docker volume` com o nome do driver Astra Trident especificado quando necessário.

Crie um volume

- Crie um volume com um driver usando o nome padrão:

```
docker volume create -d netapp --name firstVolume
```

- Criar um volume com uma instância específica do Astra Trident:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

Se você não especificar nenhum "opções", os padrões para o driver serão usados.

- Substituir o tamanho de volume predefinido. Veja o exemplo a seguir para criar um volume 20GiB com um driver:

```
docker volume create -d netapp --name my_vol --opt size=20G
```

Os tamanhos de volume são expressos como strings contendo um valor inteiro com unidades opcionais (exemplo: 10g, 20GB, 3TiB). Se nenhuma unidade for especificada, o padrão é G. as unidades de tamanho podem ser expressas como potências de 2 (B, KiB, MIB, GiB, TIB) ou potências de 10 (B, KB, MB, GB, TB). As unidades shorthand usam poderes de 2 (G GiB, T TIB,…).

Remova um volume

- Remova o volume como qualquer outro volume do Docker:

```
docker volume rm firstVolume
```

Ao utilizar o `solidfire-san` controlador, o exemplo acima elimina e elimina o volume.

Execute as etapas abaixo para atualizar o Astra Trident para Docker.

Clonar um volume

Ao usar o `ontap-nas`, `ontap-san`, , `solidfire-san` e `gcp-cvs storage drivers`, o Astra Trident pode clonar volumes. Ao usar os `ontap-nas-flexgroup` drivers ou `ontap-nas-economy`, a clonagem não é suportada. Criar um novo volume a partir de um volume existente resultará na criação de um novo instantâneo.

- Inspecione o volume para enumerar instantâneos:

```
docker volume inspect <volume_name>
```

- Crie um novo volume a partir de um volume existente. Isso resultará na criação de um novo snapshot:

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume>
```

- Criar um novo volume a partir de um instantâneo existente em um volume. Isso não criará um novo snapshot:

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

Exemplo

```
docker volume inspect firstVolume

[
    {
        "Driver": "ontap-nas",
        "Labels": null,
        "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
        "Name": "firstVolume",
        "Options": {},
        "Scope": "global",
        "Status": {
            "Snapshots": [
                {
                    "Created": "2017-02-10T19:05:00Z",
                    "Name": "hourly.2017-02-10_1505"
                }
            ]
        }
    }
]

docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume

docker volume rm clonedVolume
docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume
-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

docker volume rm volFromSnap
```

Acesse volumes criados externamente

Você pode acessar dispositivos de bloco criados externamente (ou seus clones) por contentores usando Trident **only** se eles não tiverem partições e se seu sistema de arquivos for suportado pelo Astra Trident (por exemplo: Um `ext4`-formatado `/dev/sdc1` não será acessível via Astra Trident).

Opções de volume específicas do condutor

Cada driver de armazenamento tem um conjunto diferente de opções, que você pode especificar no momento da criação do volume para personalizar o resultado. Veja abaixo as opções que se aplicam ao sistema de armazenamento configurado.

Usar essas opções durante a operação de criação de volume é simples. Forneça a opção e o valor usando o `-o` operador durante a operação CLI. Estes substituem quaisquer valores equivalentes do arquivo de

configuração JSON.

Opções de volume ONTAP

As opções de criação de volume para NFS e iSCSI incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, padrão é 1 GiB.
`spaceReserve`	Provisionamento fino ou espesso do volume, o padrão é fino. Os valores válidos são `none` (thin Provisioning) e `volume` (thick provisioned).
`snapshotPolicy`	Isto irá definir a política de instantâneos para o valor pretendido. O padrão é `none`, o que significa que nenhum instantâneo será criado automaticamente para o volume. A menos que seja modificada pelo administrador de storage, existe uma política chamada "padrão" em todos os sistemas ONTAP, que cria e retém seis snapshots por hora, dois por dia e dois por semana. Os dados preservados em um snapshot podem ser recuperados navegando para `.snapshot` o diretório em qualquer diretório do volume.
`snapshotReserve`	Isto irá definir a reserva de instantâneos para a percentagem pretendida. O padrão não é nenhum valor, o que significa que o ONTAP selecionará o snapshotServe (geralmente 5%) se você selecionou uma política de snapshotPolicy, ou 0% se a política de snapshotPolicy não for nenhuma. Você pode definir o valor padrão snapshotServe no arquivo de configuração para todos os backends ONTAP, e você pode usá-lo como uma opção de criação de volume para todos os backends ONTAP, exceto ONTAP-nas-economy.
`splitOnClone`	Ao clonar um volume, isso fará com que o ONTAP divida imediatamente o clone de seu pai. A predefinição é `false`. Alguns casos de uso para clonagem de volumes são melhor servidos dividindo o clone de seu pai imediatamente após a criação, porque é improvável que haja alguma oportunidade de eficiência de storage. Por exemplo, clonar um banco de dados vazio pode oferecer grande economia de tempo, mas pouca economia de armazenamento, por isso é melhor dividir o clone imediatamente.

Opção	Descrição
`encryption`	Ative a criptografia de volume do NetApp (NVE) no novo volume; o padrão é `false`. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se o NAE estiver ativado no back-end, qualquer volume provisionado no Astra Trident será o NAE ativado. Para obter mais informações, consulte: "Como o Astra Trident funciona com NVE e NAE".
`tieringPolicy`	Define a política de disposição em categorias a ser usada para o volume. Isso decide se os dados são movidos para a categoria de nuvem quando ficam inativos (frios).

As seguintes opções adicionais são para NFS **somente**:

Opção	Descrição
`unixPermissions`	Isso controla o conjunto de permissões para o próprio volume. Por padrão, as permissões serão definidas como `---rwxr-xr-x, ou em notação numérica 0755, e `root` serão o proprietário. O texto ou o formato numérico funcionarão.
`snapshotDir`	Definir isso como `true` tornará o `.snapshot` diretório visível para os clientes que acessam o volume. O valor padrão é `false`, o que significa que a visibilidade `.snapshot` do diretório está desativada por padrão. Algumas imagens, por exemplo, a imagem oficial do MySQL, não funcionam como esperado quando o `.snapshot` diretório está visível.
`exportPolicy`	Define a política de exportação a ser utilizada para o volume. A predefinição é `default`.
`securityStyle`	Define o estilo de segurança a ser usado para acesso ao volume. A predefinição é `unix`. Os valores válidos são `unix` e `mixed`.

As seguintes opções adicionais são para iSCSI **somente**:

Opção	Descrição
`fileSystemType`	Define o sistema de ficheiros utilizado para formatar volumes iSCSI. A predefinição é `ext4`. Os valores válidos são `ext3`, `ext4`, e `xfs`.
`spaceAllocation`	Definir esta opção como `false` desativa a funcionalidade de alocação de espaço do LUN. O valor padrão é `true`, o que significa que o ONTAP notifica o host quando o volume ficou sem espaço e o LUN no volume não pode aceitar gravações. Essa opção também permite que o ONTAP recupere espaço automaticamente quando o host exclui dados.

Exemplos

Veja os exemplos abaixo:

- Criar um volume 10GiBD:

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Criar um volume 100GiBD com instantâneos:

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- Crie um volume que tenha o bit setuid ativado:

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

O tamanho mínimo do volume é 20MiB.

Se a reserva de snapshot não for especificada e a política de snapshot for `none`, o Trident usará uma reserva de snapshot de 0%.

- Criar um volume sem política de snapshot e sem reserva de snapshot:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Crie um volume sem política de snapshot e uma reserva de snapshot personalizada de 10%:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Crie um volume com uma política de snapshot e uma reserva de snapshot personalizada de 10%:

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- Crie um volume com uma política de snapshot e aceite a reserva de snapshot padrão do ONTAP (geralmente 5%):

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Opções de volume do software Element

As opções de software Element expõem as políticas de tamanho e qualidade do serviço (QoS) associadas ao volume. Quando o volume é criado, a política de QoS associada a ele é especificada usando a `-o type=service_level` nomenclatura.

A primeira etapa para definir um nível de serviço QoS com o driver Element é criar pelo menos um tipo e especificar o IOPS mínimo, máximo e de pico associado a um nome no arquivo de configuração.

Outras opções de criação de volume de software Element incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, padrão para 1GiB ou entrada de configuração … "Padrões": 5G.
`blocksize`	Use 512 ou 4096, o padrão é 512 ou a entrada de configuração DefaultBlockSize.

Exemplo

Veja o seguinte arquivo de configuração de exemplo com definições de QoS:

```
{
    "...": "..."
    "Types": [
        {
            "Type": "Bronze",
            "Qos": {
                "minIOPS": 1000,
                "maxIOPS": 2000,
                "burstIOPS": 4000
            }
        },
        {
            "Type": "Silver",
            "Qos": {
                "minIOPS": 4000,
                "maxIOPS": 6000,
                "burstIOPS": 8000
            }
        },
        {
            "Type": "Gold",
            "Qos": {
                "minIOPS": 6000,
                "maxIOPS": 8000,
                "burstIOPS": 10000
            }
        }
    ]
}
```

Na configuração acima, temos três definições de política: Bronze, prata e ouro. Esses nomes são arbitrários.

- Criar um volume 10GiB Gold:

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Criar um volume Bronze 100GiB:

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

Recolher registos

Você pode coletar Registros para obter ajuda com a solução de problemas. O método que você usa para coletar os logs varia de acordo com a forma como você está executando o plugin Docker.

Recolha registos para resolução de problemas

Passos

1. Se você estiver executando o Astra Trident usando o método de plug-in gerenciado recomendado (ou seja, usando `docker plugin` comandos), visualize-os da seguinte forma:

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
journalctl -u docker | grep 4fb97d2b956b
```

   O nível de registo padrão deve permitir diagnosticar a maioria dos problemas. Se você achar que isso não é suficiente, você pode ativar o Registro de depuração.

2. Para ativar o registo de depuração, instale o plug-in com o registo de depuração ativado:

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

   Ou, ative o registo de depuração quando o plug-in já estiver instalado:

```
docker plugin disable <plugin>
docker plugin set <plugin> debug=true
docker plugin enable <plugin>
```

3. Se você estiver executando o binário em si no host, os logs estarão disponíveis no diretório do host `/var/log/netappdvp`. Para ativar o registo de depuração, especifique `-debug` quando executar o plugin.

Dicas gerais de solução de problemas

- O problema mais comum em que novos usuários são executados é uma configuração incorreta que impede que o plugin seja inicializado. Quando isso acontecer, você provavelmente verá uma mensagem como esta quando você tentar instalar ou ativar o plugin:

  ```
  Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
  connect: no such file or directory
  ```

Isto significa que o plugin falhou ao iniciar. Felizmente, o plugin foi construído com uma capacidade de Registro abrangente que deve ajudá-lo a diagnosticar a maioria dos problemas que você provavelmente encontrará.

- Se houver problemas com a montagem de um PV em um recipiente, certifique-se de que `rpcbind` está instalado e funcionando. Use o gerenciador de pacotes necessário para o sistema operacional do host e verifique se `rpcbind` está em execução. Você pode verificar o status do serviço rpcbind executando um `systemctl status rpcbind` ou seu equivalente.

Gerenciar várias instâncias do Astra Trident

Várias instâncias do Trident são necessárias quando você deseja ter várias configurações de storage disponíveis simultaneamente. A chave para várias instâncias é dar nomes diferentes usando a `--alias` opção com o plug-in em contentor, ou `--volume-driver` opção ao instanciar o Trident no host.

Etapas para o plugin gerenciado do Docker (versão 1,13/17,03 ou posterior)

1. Inicie a primeira instância especificando um alias e um arquivo de configuração.

```
docker plugin install --grant-all-permissions --alias silver
netapp/trident-plugin:21.07 config=silver.json
```

2. Inicie a segunda instância, especificando um alias diferente e arquivo de configuração.

```
docker plugin install --grant-all-permissions --alias gold
netapp/trident-plugin:21.07 config=gold.json
```

3. Crie volumes especificando o alias como o nome do driver.

   Por exemplo, para o volume de ouro:

```
docker volume create -d gold --name ntapGold
```

   Por exemplo, para o volume prateado:

```
docker volume create -d silver --name ntapSilver
```

Passos para o tradicional (versão 1,12 ou anterior)

1. Inicie o plugin com uma configuração NFS usando um ID de driver personalizado:

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. Inicie o plug-in com uma configuração iSCSI usando um ID de driver personalizado:

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. Provisione volumes Docker para cada instância de driver:

   Por exemplo, para NFS:

```
docker volume create -d netapp-nas --name my_nfs_vol
```

   Por exemplo, para iSCSI:

```
docker volume create -d netapp-san --name my_iscsi_vol
```

Opções de configuração de armazenamento

Consulte as opções de configuração disponíveis para suas configurações do Astra Trident.

Opções de configuração global

Essas opções de configuração se aplicam a todas as configurações do Astra Trident, independentemente da plataforma de storage usada.

Opção	Descrição	Exemplo
`version`	Número da versão do ficheiro de configuração	`1`
`storageDriverName`	Nome do driver de armazenamento	`ontap-nas` `ontap-san`, , `ontap-nas-economy` `ontap-nas-flexgroup` , , , `solidfire-san`
`storagePrefix`	Prefixo opcional para nomes de volume. Padrão: `netappdvp_`.	`staging_`

Opção	Descrição	Exemplo
`limitVolumeSize`	Restrição opcional nos tamanhos de volume. Padrão: "" (não aplicado)	`10g`

Não use `storagePrefix` (incluindo o padrão) para backends de elemento. Por padrão, o `solidfire-san` driver ignorará essa configuração e não usará um prefixo. Recomendamos usar um tenantID específico para mapeamento de volume do Docker ou usar os dados de atributo que são preenchidos com a versão do Docker, informações de driver e nome bruto do Docker nos casos em que qualquer nome munging pode ter sido usado.

As opções padrão estão disponíveis para evitar ter que especificá-las em cada volume criado. A `size` opção está disponível para todos os tipos de controlador. Consulte a seção Configuração do ONTAP para obter um exemplo de como definir o tamanho padrão do volume.

Opção	Descrição	Exemplo
`size`	Tamanho padrão opcional para novos volumes. Predefinição: `1G`	`10G`

Configuração ONTAP

Além dos valores de configuração global acima, ao usar o ONTAP, as seguintes opções de nível superior estão disponíveis.

Opção	Descrição	Exemplo
`managementLIF`	Endereço IP do ONTAP Management LIF. Você pode especificar um nome de domínio totalmente qualificado (FQDN).	`10.0.0.1`

Opção	Descrição	Exemplo
`dataLIF`	Endereço IP do protocolo LIF. • ONTAP nas drivers*: Recomendamos especificar `dataLIF`. Se não for fornecido, o Astra Trident obtém LIFs de dados do SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) a ser usado para as operações de montagem NFS, permitindo que você crie um DNS de round-robin para balanceamento de carga em vários LIFs de dados. **Drivers SAN ONTAP**: Não especifique para iSCSI. O Astra Trident usa "Mapa de LUN seletivo da ONTAP" para descobrir os LIFs iSCI necessários para estabelecer uma sessão de vários caminhos. Um aviso é gerado se `dataLIF` for definido explicitamente.	`10.0.0.2`
`svm`	Máquina virtual de armazenamento a utilizar (necessária, se o LIF de gestão for um LIF de cluster)	`svm_nfs`
`username`	Nome de utilizador para ligar ao dispositivo de armazenamento	`vsadmin`
`password`	Palavra-passe para ligar ao dispositivo de armazenamento	`secret`
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído ao SVM). Para `ontap-nas-flexgroup` o driver, essa opção é ignorada. Todos os agregados atribuídos ao SVM são usados para provisionar um volume FlexGroup.	`aggr1`
`limitAggregateUsage`	Opcional, falha no provisionamento se o uso estiver acima dessa porcentagem	`75%`

Opção	Descrição	Exemplo
`nfsMountOptions`	Controle refinado das opções de montagem NFS; o padrão é "-o nfsvers 3". **Disponível apenas para os `ontap-nas` condutores e `ontap-nas-economy`.** "Consulte as informações de configuração do host NFS aqui".	`-o nfsvers=4`
`igroupName`	O Astra Trident cria e gerencia por nó `igroups` `netappdvp` como . Este valor não pode ser alterado ou omitido. **Disponível apenas para `ontap-san` o condutor.**	`netappdvp`
`limitVolumeSize`	Tamanho máximo do volume requestable e tamanho do volume pai de qtree. **Para o `ontap-nas-economy` driver, essa opção limita adicionalmente o tamanho dos FlexVols que ele cria.**	`300g`
`qtreesPerFlexvol`	Qtrees máximos por FlexVol, tem de estar no intervalo [50, 300], o padrão é 200. **Para `ontap-nas-economy` o driver, esta opção permite personalizar o número máximo de qtrees por FlexVol.**	`300`
`sanType`	**Suportado apenas para `ontap-san` driver.** Utilize para selecionar `iscsi` para iSCSI ou `nvme` para NVMe/TCP.	`iscsi` se estiver em branco

As opções padrão estão disponíveis para evitar ter que especificá-las em cada volume criado:

Opção	Descrição	Exemplo
`spaceReserve`	Modo de reserva de espaço; `none` (thin Provisioning) ou `volume` (thick)	`none`
`snapshotPolicy`	Política de instantâneos a utilizar, a predefinição é `none`	`none`

Opção	Descrição	Exemplo
`snapshotReserve`	O padrão é "" para aceitar o padrão ONTAP	`10`
`splitOnClone`	Divida um clone de seu pai na criação, o padrão é `false`	`false`
`encryption`	Ativa a criptografia de volume NetApp (NVE) no novo volume; o padrão é `false`. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se o NAE estiver ativado no back-end, qualquer volume provisionado no Astra Trident será o NAE ativado. Para obter mais informações, consulte: "Como o Astra Trident funciona com NVE e NAE".	verdadeiro
`unixPermissions`	Opção nas para volumes NFS provisionados, o padrão é `777`	`777`
`snapshotDir`	Opção nas para acesso ao `.snapshot` diretório, o padrão é `false`	`true`
`exportPolicy`	A opção nas para a política de exportação NFS a usar, o padrão é `default`	`default`
`securityStyle`	Opção nas para acesso ao volume NFS provisionado. Estilos de segurança e `unix` suporte de NFS `mixed`. A predefinição é `unix`.	`unix`
`fileSystemType`	Opção SAN para selecionar o tipo de sistema de arquivos, o padrão é `ext4`	`xfs`
`tieringPolicy`	A política de disposição em categorias a usar, o padrão é `none`; `snapshot-only` para a configuração pré-ONTAP 9.5 SVM-DR	`none`

Opções de dimensionamento

Os `ontap-nas` drivers e `ontap-san` criam um ONTAP FlexVol para cada volume do Docker. O ONTAP dá suporte a até 1000 FlexVols por nó de cluster com um máximo de cluster de 12.000 FlexVols. Se os requisitos de volume do Docker se ajustarem a essa limitação, `ontap-nas` o driver será a solução nas preferida devido aos recursos adicionais oferecidos pelo FlexVols, como snapshots Docker volume granular e clonagem.

Se você precisar de mais volumes do Docker do que pode ser acomodado pelos limites do FlexVol, escolha o `ontap-nas-economy` ou o `ontap-san-economy` driver.

`ontap-nas-economy`O driver cria volumes do Docker como Qtrees do ONTAP em um pool de FlexVols gerenciados automaticamente. As Qtrees oferecem dimensionamento muito maior, até 100.000 PB por nó de cluster e 2.400.000 PB por cluster, à custa de alguns recursos. `ontap-nas-economy`O driver não oferece suporte a snapshots ou clonagem granular de volume do Docker.

No momento, o `ontap-nas-economy` driver não é compatível com o Docker Swarm, porque o Swarm não orquestra a criação de volume em vários nós.

`ontap-san-economy`O driver cria volumes do Docker como LUNs ONTAP em um pool compartilhado de FlexVols gerenciados automaticamente. Dessa forma, cada FlexVol não se restringe a apenas um LUN e oferece melhor escalabilidade para workloads SAN. Dependendo do storage array, o ONTAP oferece suporte para até 16384 LUNs por cluster. Como os volumes são LUNs abaixo, esse driver oferece suporte a snapshots e clonagem granular do Docker volume.

Escolha o `ontap-nas-flexgroup` driver para aumentar o paralelismo para um único volume que pode crescer para o intervalo de petabytes com bilhões de arquivos. Alguns casos de uso ideais para FlexGroups incluem IA/ML/DL, big data e análise, compilações de software, streaming, repositórios de arquivos e assim por diante. O Trident usa todos os agregados atribuídos a uma SVM ao provisionar um volume FlexGroup. O suporte do FlexGroup no Trident também tem as seguintes considerações:

- Requer ONTAP versão 9,2 ou superior.
- A partir desta redação, FlexGroups só suportam NFS v3.
- Recomendado para ativar os identificadores NFSv3 de 64 bits para o SVM.
- O tamanho mínimo recomendado de membro/volume FlexGroup é 100GiB.
- A clonagem não é compatível com volumes FlexGroup.

Para obter informações sobre FlexGroups e cargas de trabalho apropriadas para FlexGroups, consulte "Guia de práticas recomendadas e implementação de volumes do NetApp FlexGroup".

Para obter recursos avançados e grande escala no mesmo ambiente, você pode executar várias instâncias do Docker volume Plugin, com uma usando `ontap-nas` e outra usando `ontap-nas-economy`o .

Exemplo de arquivos de configuração do ONTAP

Exemplo de NFS para o driver <code> ONTAP-nas</code>

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1",
    "defaults": {
      "size": "10G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Exemplo de NFS para o driver <code> ONTAP-nas-FlexGroup </code>

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "defaults": {
      "size": "100G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Exemplo de NFS para o driver <code> ONTAP-nas-economy</code>

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1"
}
```

Exemplo iSCSI para o controlador <code> ONTAP-san</code>

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1",
    "igroupName": "netappdvp"
}
```

Exemplo de NFS para o driver <code> ONTAP-San-economy</code>

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi_eco",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1",
    "igroupName": "netappdvp"
}
```

Exemplo de NVMe/TCP para o driver <code> ONTAP-san</code>

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username":"vsadmin",
  "password":"password",
  "sanType": "nvme",
  "useREST": true
}
```

Configuração do software Element

Além dos valores de configuração global, ao usar o software Element (NetApp HCI/SolidFire), essas opções estão disponíveis.

Opção	Descrição	Exemplo
`Endpoint`	/<login>:<password>/<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	Endereço IP iSCSI e porta	10,0.0,7:3260
`TenantName`	Locatário do SolidFireF para usar (criado se não for encontrado)	`docker`
`InitiatorIFace`	Especifique a interface ao restringir o tráfego iSCSI a uma interface não predefinida	`default`
`Types`	Especificações de QoS	Veja o exemplo abaixo
`LegacyNamePrefix`	Prefixo para instalações Trident atualizadas. Se você usou uma versão do Trident anterior a 1.3.2 e executar uma atualização com volumes existentes, precisará definir esse valor para acessar seus volumes antigos que foram mapeados pelo método de nome de volume.	`netappdvp-`

O `solidfire-san` driver não suporta Docker Swarm.

Exemplo de arquivo de configuração de software Element

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
    "SVIP": "10.0.0.7:3260",
    "TenantName": "docker",
    "InitiatorIFace": "default",
    "Types": [
        {
            "Type": "Bronze",
            "Qos": {
                "minIOPS": 1000,
                "maxIOPS": 2000,
                "burstIOPS": 4000
            }
        },
        {
            "Type": "Silver",
            "Qos": {
                "minIOPS": 4000,
                "maxIOPS": 6000,
                "burstIOPS": 8000
            }
        },
        {
            "Type": "Gold",
            "Qos": {
                "minIOPS": 6000,
                "maxIOPS": 8000,
                "burstIOPS": 10000
            }
        }
    ]
}
```

Problemas e limitações conhecidos

Encontre informações sobre problemas e limitações conhecidos ao usar o Astra Trident com Docker.

A atualização do plug-in de volume do Docker do Trident para 20,10 e posterior a partir de versões mais antigas resulta em falha de atualização com o erro de nenhum arquivo ou diretório.

Solução alternativa

1. Desative o plugin.

```
docker plugin disable -f netapp:latest
```

2. Remova o plugin.

```
docker plugin rm -f netapp:latest
```

3. Reinstale o plugin fornecendo o parâmetro extra `config`.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

Os nomes dos volumes devem ter um mínimo de 2 carateres.

Esta é uma limitação de cliente Docker. O cliente interpretará um único nome de caractere como sendo um caminho do Windows. "Veja o bug 25773".

O Docker Swarm tem certos comportamentos que impedem o Astra Trident de dar suporte a ele em cada combinação de storage e driver.

- Docker Swarm atualmente faz uso do nome do volume em vez de ID do volume como seu identificador de volume exclusivo.
- As solicitações de volume são enviadas simultaneamente para cada nó em um cluster Swarm.
- Os plug-ins de volume (incluindo o Astra Trident) devem ser executados de forma independente em cada nó em um cluster do Swarm. Devido à forma como o ONTAP funciona e como os `ontap-nas` drivers e `ontap-san` funcionam, eles são os únicos que podem operar dentro dessas limitações.

O resto dos pilotos estão sujeitos a problemas como condições de corrida que podem resultar na criação de um grande número de volumes para uma única solicitação sem um claro "vencedor"; por exemplo, o elemento tem um recurso que permite que os volumes tenham o mesmo nome, mas IDs diferentes.

O NetApp forneceu feedback à equipe do Docker, mas não tem qualquer indicação de recurso futuro.

Se um FlexGroup estiver sendo provisionado, o ONTAP não provisiona um segundo FlexGroup se o segundo FlexGroup tiver um ou mais agregados em comum com o FlexGroup sendo provisionado.

Práticas recomendadas e recomendações

Implantação

Use as recomendações listadas aqui quando implantar o Astra Trident.

Implante em um namespace dedicado

"Namespaces" fornecer separação administrativa entre diferentes aplicações e são uma barreira para o compartilhamento de recursos. Por exemplo, um PVC de um namespace não pode ser consumido de outro. O Astra Trident fornece recursos PV para todos os namespaces no cluster do Kubernetes e, consequentemente, utiliza uma conta de serviço que elevou o Privileges.

Além disso, o acesso ao pod Trident pode permitir que um usuário acesse credenciais do sistema de storage e outras informações confidenciais. É importante garantir que os usuários de aplicativos e aplicativos de gerenciamento não tenham a capacidade de acessar as definições de objetos do Trident ou os próprios pods.

Use cotas e limites de intervalo para controlar o consumo de armazenamento

O Kubernetes tem dois recursos que, quando combinados, fornecem um mecanismo avançado para limitar o consumo de recursos pelas aplicações. O "mecanismo de cota de storage" permite que o administrador implemente limites de consumo globais e específicos de classe de storage, de contagem de objetos e capacidade em uma base por namespace. Além disso, o uso de a "limite de alcance" garante que as solicitações de PVC estejam dentro de um valor mínimo e máximo antes que a solicitação seja encaminhada para o provisionador.

Esses valores são definidos em uma base por namespace, o que significa que cada namespace deve ter valores definidos que se encaixam em seus requisitos de recursos. Consulte aqui para obter informações "como alavancar cotas" sobre .

Configuração de armazenamento

Cada plataforma de storage do portfólio do NetApp tem funcionalidades exclusivas que beneficiam aplicações, em contêineres ou não.

Visão geral da plataforma

O Trident funciona com ONTAP e Element. Não há uma plataforma que seja mais adequada para todos os aplicativos e cenários do que outra, no entanto, as necessidades do aplicativo e da equipe que administra o dispositivo devem ser levadas em conta ao escolher uma plataforma.

Você deve seguir as práticas recomendadas de linha de base para o sistema operacional host com o protocolo que você está utilizando. Opcionalmente, você pode considerar a incorporação de práticas recomendadas de aplicativos, quando disponíveis, com configurações de backend, classe de armazenamento e PVC para otimizar o armazenamento para aplicativos específicos.

Práticas recomendadas de ONTAP e Cloud Volumes ONTAP

Conheça as práticas recomendadas para configurar o ONTAP e o Cloud Volumes ONTAP for Trident.

As recomendações a seguir são diretrizes para configuração do ONTAP para workloads em contêineres, que

consomem volumes provisionados dinamicamente pelo Trident. Cada um deve ser considerado e avaliado quanto à adequação em seu ambiente.

Use SVM(s) dedicados ao Trident

As máquinas virtuais de storage (SVMs) fornecem isolamento e separação administrativa entre locatários em um sistema ONTAP. A dedicação de um SVM a aplicações permite a delegação do Privileges e permite aplicar práticas recomendadas para limitar o consumo de recursos.

Há várias opções disponíveis para o gerenciamento do SVM:

- Fornecer a interface de gerenciamento de cluster na configuração de back-end, juntamente com as credenciais apropriadas, e especificar o nome da SVM.
- Crie uma interface de gerenciamento dedicada ao SVM com o Gerenciador de sistemas do ONTAP ou a CLI.
- Compartilhe a função de gerenciamento com uma interface de dados NFS.

Em cada caso, a interface deve estar em DNS, e o nome DNS deve ser usado ao configurar o Trident. Isso ajuda a facilitar alguns cenários de DR, por exemplo, SVM-DR sem o uso de retenção de identidade de rede.

No entanto, não há preferência entre ter um LIF de gerenciamento dedicado ou compartilhado para o SVM, você deve garantir que suas políticas de segurança de rede estejam alinhadas com a abordagem escolhida. Independentemente disso, o LIF de gerenciamento deve ser acessível via DNS para facilitar a máxima flexibilidade deve "SVM-DR" ser usado em conjunto com o Trident.

Limite a contagem máxima de volume

Os sistemas de storage ONTAP têm uma contagem de volume máxima, que varia de acordo com a versão do software e a plataforma de hardware. "NetApp Hardware Universe"Consulte para obter informações sobre a sua plataforma específica e a versão do ONTAP para determinar os limites exatos. Quando a contagem de volume está esgotada, as operações de provisionamento falham não apenas para o Trident, mas para todas as solicitações de storage.

Os Trident `ontap-nas` e `ontap-san` os drivers provisionam um Flexvolume para cada volume persistente (PV) do Kubernetes criado. O `ontap-nas-economy` driver cria aproximadamente um Flexvolume para cada 200 PVS (configurável entre 50 e 300). O `ontap-san-economy` driver cria aproximadamente um Flexvolume para cada 100 PVS (configurável entre 50 e 200). Para evitar que o Trident consuma todos os volumes disponíveis no sistema de storage, defina um limite para o SVM. Você pode fazer isso a partir da linha de comando:

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

O valor para `max-volumes` varia com base em vários critérios específicos para o seu ambiente:

- O número de volumes existentes no cluster do ONTAP
- O número de volumes que você espera provisionar fora do Trident para outras aplicações
- O número de volumes persistentes esperado para ser consumido pelas aplicações Kubernetes

O `max-volumes` valor é o total de volumes provisionados em todos os nós do cluster ONTAP e não em um nó ONTAP individual. Como resultado, você pode encontrar algumas condições em que um nó de cluster do ONTAP pode ter muito mais ou menos volumes provisionados pelo Trident do que outro nó.

Por exemplo, um cluster ONTAP de dois nós tem a capacidade de hospedar um máximo de 2000 volumes flexíveis. Ter a contagem de volume máxima definida para 1250 parece muito razoável. No entanto, se apenas "agregados" de um nó forem atribuídos à SVM, ou se os agregados atribuídos de um nó não puderem ser provisionados (por exemplo, devido à capacidade), o outro nó se tornará o destino de todos os volumes provisionados pelo Trident. Isso significa que o limite de volume pode ser alcançado para esse nó antes que o `max-volumes` valor seja atingido, resultando em impacto no Trident e em outras operações de volume que usam esse nó. **Você pode evitar essa situação garantindo que os agregados de cada nó no cluster sejam atribuídos ao SVM usado pelo Trident em números iguais.**

Limite o tamanho máximo de volumes criados pelo Trident

Para configurar o tamanho máximo para volumes que podem ser criados pelo Trident, use o `limitVolumeSize` parâmetro em `backend.json` sua definição.

Além de controlar o tamanho do volume no storage array, você também deve utilizar os recursos do Kubernetes.

Configure o Trident para usar o CHAP bidirecional

Você pode especificar o iniciador CHAP e os nomes de usuário e senhas de destino em sua definição de back-end e ter o Trident Enable CHAP no SVM. Usando o `useCHAP` parâmetro em sua configuração de back-end, o Trident autentica conexões iSCSI para backends ONTAP com CHAP.

Criar e usar uma política de QoS SVM

A utilização de uma política de QoS ONTAP aplicada à SVM limita o número de consumíveis de IOPS pelos volumes provisionados pelo Trident. Isso ajuda "evite um bully" a evitar que o volume fora de controle afete workloads fora do SVM do Trident.

Você pode criar uma política de QoS para o SVM em algumas etapas. Consulte a documentação da sua versão do ONTAP para obter as informações mais precisas. O exemplo abaixo cria uma política de QoS que limita o total de IOPS disponível para o SVM a 5000.

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

Além disso, se a sua versão do ONTAP for compatível com ela, considere o uso de um mínimo de QoS para garantir uma taxa de transferência para workloads em contêineres. A QoS adaptável não é compatível com uma política de nível SVM.

O número de IOPS dedicado aos workloads em contêineres depende de muitos aspectos. Entre outras coisas, estas incluem:

- Outros workloads que usam o storage array. Se houver outras cargas de trabalho, não relacionadas à implantação do Kubernetes, utilizando os recursos de storage, deve-se tomar cuidado para garantir que essas cargas de trabalho não sejam acidentalmente afetadas.

- Workloads esperados em execução em contêineres. Se os workloads com requisitos de IOPS altos forem executados em contêineres, uma política de QoS baixa resulta em uma experiência ruim.

É importante lembrar que uma política de QoS atribuída no nível SVM resulta em todos os volumes provisionados ao SVM que compartilham o mesmo pool de IOPS. Se uma, ou um número pequeno, das aplicações em contêiner tiverem um requisito de IOPS alto, isso pode se tornar um bully para os outros workloads em contêiner. Se esse for o caso, você pode considerar o uso de automação externa para atribuir políticas de QoS por volume.

> Você deve atribuir o grupo de políticas de QoS ao SVM **somente** se a versão do ONTAP for anterior a 9,8.

Criar grupos de política de QoS para Trident

A qualidade do serviço (QoS) garante que a performance de workloads essenciais não é degradada pelos workloads da concorrência. Os grupos de política de QoS do ONTAP fornecem opções de QoS para volumes e permitem que os usuários definam o limite máximo de taxa de transferência para um ou mais workloads. Para obter mais informações sobre QoS, "Garantir taxa de transferência com QoS" consulte . É possível especificar grupos de políticas de QoS no back-end ou em um pool de storage, e eles são aplicados a cada volume criado nesse pool ou back-end.

O ONTAP tem dois tipos de grupos de política de QoS: Tradicional e adaptável. Os grupos de políticas tradicionais fornecem uma taxa de transferência máxima fixa (ou mínima, em versões posteriores) em IOPS. O serviço adaptável dimensiona automaticamente a taxa de transferência para o tamanho do workload, mantendo a taxa de IOPS para TBs|GBs conforme o tamanho do workload muda. Isso proporciona uma vantagem significativa ao gerenciar centenas ou milhares de workloads em uma implantação grande.

Considere o seguinte ao criar grupos de política de QoS:

- Você deve definir a `qosPolicy` chave no `defaults` bloco da configuração de back-end. Veja o seguinte exemplo de configuração de back-end:

```
  ---
version: 1
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
- labels:
    performance: extreme
  defaults:
    adaptiveQosPolicy: extremely-adaptive-pg
- labels:
    performance: premium
  defaults:
    qosPolicy: premium-pg
```

- Você deve aplicar os grupos de políticas por volume, para que cada volume obtenha toda a taxa de transferência, conforme especificado pelo grupo de políticas. Grupos de políticas compartilhadas não são suportados.

Para obter mais informações sobre grupos de políticas de QoS, "Comandos de QoS ONTAP 9.8" consulte .

Limitar o acesso a recursos de storage aos membros do cluster do Kubernetes

Limitar o acesso aos volumes NFS e iSCSI LUNs criados pelo Trident é um componente essencial da postura de segurança para a implantação do Kubernetes. Isso impede que os hosts que não fazem parte do cluster do Kubernetes acessem os volumes e potencialmente modifiquem os dados inesperadamente.

É importante entender que os namespaces são o limite lógico dos recursos no Kubernetes. A suposição é que os recursos no mesmo namespace são capazes de ser compartilhados, no entanto, é importante, não há capacidade entre namespace. Isso significa que, embora os PVS sejam objetos globais, quando vinculados a um PVC, eles só são acessíveis por pods que estão no mesmo namespace. **É fundamental garantir que os namespaces sejam usados para fornecer separação quando apropriado.**

A principal preocupação da maioria das organizações com relação à segurança de dados em um contexto do Kubernetes é que um processo em um contêiner pode acessar o storage montado no host, mas que não se destina ao contêiner. "Namespaces" foram concebidos para evitar este tipo de compromisso. No entanto, há uma exceção: Contentores privilegiados.

Um contentor privilegiado é aquele que é executado com permissões substancialmente mais no nível do host do que o normal. Estes não são negados por padrão, portanto, certifique-se de desativar a capacidade "diretivas de segurança do pod" usando o .

Para volumes em que o acesso é desejado tanto do Kubernetes quanto de hosts externos, o storage deve ser gerenciado de maneira tradicional, com o PV introduzido pelo administrador e não gerenciado pelo Trident. Isso garante que o volume de storage seja destruído somente quando o Kubernetes e os hosts externos forem

desconetados e não estiverem mais usando o volume. Além disso, é possível aplicar uma política de exportação personalizada, que permite o acesso dos nós de cluster do Kubernetes e dos servidores direcionados fora do cluster do Kubernetes.

Para implantações com nós de infraestrutura dedicados (por exemplo, OpenShift) ou outros nós que não conseguem programar aplicativos de usuário, políticas de exportação separadas devem ser usadas para limitar ainda mais o acesso aos recursos de storage. Isso inclui a criação de uma política de exportação para serviços que são implantados nesses nós de infraestrutura (por exemplo, os serviços de métricas e Registro OpenShift) e aplicativos padrão que são implantados em nós que não são de infraestrutura.

Use uma política de exportação dedicada

Você deve garantir que existe uma política de exportação para cada back-end que permita somente o acesso aos nós presentes no cluster do Kubernetes. O Trident pode criar e gerenciar automaticamente políticas de exportação. Dessa forma, o Trident limita o acesso aos volumes provisionados por TI aos nós no cluster do Kubernetes e simplifica a adição/exclusão de nós.

Como alternativa, você também pode criar uma política de exportação manualmente e preenchê-la com uma ou mais regras de exportação que processam cada solicitação de acesso de nó:

- Use o `vserver export-policy create` comando ONTAP CLI para criar a política de exportação.
- Adicione regras à política de exportação usando o `vserver export-policy rule create` comando ONTAP CLI.

Executar esses comandos permite restringir quais nós do Kubernetes têm acesso aos dados.

`showmount`Desativar o SVM da aplicação

O `showmount` recurso permite que um cliente NFS consulte o SVM para obter uma lista de exportações de NFS disponíveis. Um pod implantado no cluster do Kubernetes pode emitir o `showmount -e` comando contra o LIF de dados e receber uma lista de montagens disponíveis, incluindo aquelas às quais ele não tem acesso. Embora isso, por si só, não seja um compromisso de segurança, ele fornece informações desnecessárias potencialmente ajudando um usuário não autorizado a se conetar a uma exportação NFS.

Você deve desativar `showmount` usando o comando ONTAP CLI no nível da SVM:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

Práticas recomendadas da SolidFire

Conheça as práticas recomendadas para configurar o armazenamento SolidFire para Trident.

Crie uma conta SolidFire

Cada conta do SolidFire representa um proprietário de volume exclusivo e recebe seu próprio conjunto de credenciais do Protocolo de Autenticação de desafio-aperto (CHAP). Você pode acessar volumes atribuídos a uma conta usando o nome da conta e as credenciais CHAP relativas ou por meio de um grupo de acesso de volume. Uma conta pode ter até dois mil volumes atribuídos a ela, mas um volume pode pertencer a apenas uma conta.

Crie uma política de QoS

Use as políticas de qualidade do serviço (QoS) do SolidFire se quiser criar e salvar uma configuração padronizada de qualidade do serviço que pode ser aplicada a muitos volumes.

Você pode definir parâmetros de QoS em uma base por volume. O desempenho de cada volume pode ser garantido definindo três parâmetros configuráveis que definem a QoS: Min IOPS, Max IOPS e Burst IOPS.

Aqui estão os possíveis valores de IOPS mínimo, máximo e de pico sazonal para o tamanho de bloco 4Kb.

Parâmetro IOPS	Definição	Valor mín	Valor padrão	Valor máximo (4Kb)
IOPS mín	O nível garantido de desempenho para um volume.	50	50	15000
IOPS máx	O desempenho não excederá este limite.	50	15000	200.000
IOPS de explosão	Máximo de IOPS permitido em um cenário de pico curto.	50	15000	200.000

Embora o IOPS máximo e o IOPS Burst possam ser definidos até 200.000 K, o desempenho máximo real de um volume é limitado pelo uso do cluster e pelo desempenho por nó.

O tamanho do bloco e a largura de banda têm uma influência direta no número de IOPS. À medida que os tamanhos de blocos aumentam, o sistema aumenta a largura de banda para um nível necessário para processar os tamanhos de blocos maiores. À medida que a largura de banda aumenta, o número de IOPS que o sistema consegue atingir diminui. "SolidFire qualidade do serviço"Consulte para obter mais informações sobre QoS e desempenho.

Autenticação SolidFire

O Element suporta dois métodos de autenticação: CHAP e volume Access Groups (VAG). O CHAP usa o protocolo CHAP para autenticar o host no back-end. Os grupos de acesso de volume controlam o acesso aos volumes que ele provisiona. O NetApp recomenda usar o CHAP para autenticação, pois é mais simples e não tem limites de escala.

O Trident com o provisionador de CSI aprimorado suporta o uso da autenticação CHAP. Os VAG só devem ser utilizados no modo de funcionamento tradicional não CSI.

A autenticação CHAP (verificação de que o iniciador é o usuário de volume pretendido) é suportada apenas com controle de acesso baseado em conta. Se você estiver usando CHAP para autenticação, duas opções estão disponíveis: CHAP unidirecional e CHAP bidirecional. O CHAP unidirecional autentica o acesso ao volume usando o nome da conta do SolidFire e o segredo do iniciador. A opção CHAP bidirecional fornece a maneira mais segura de autenticar o volume porque o volume autentica o host através do nome da conta e do segredo do iniciador e, em seguida, o host autentica o volume através do nome da conta e do segredo de destino.

No entanto, se o CHAP não puder ser ativado e os VAG forem necessários, crie o grupo de acesso e adicione

os iniciadores e volumes do host ao grupo de acesso. Cada IQN que você adicionar a um grupo de acesso pode acessar cada volume no grupo com ou sem autenticação CHAP. Se o iniciador iSCSI estiver configurado para usar autenticação CHAP, o controle de acesso baseado em conta será usado. Se o iniciador iSCSI não estiver configurado para usar a autenticação CHAP, o controle de acesso ao grupo de acesso de volume será usado.

Onde encontrar mais informações?

Alguns dos documentos de melhores práticas estão listados abaixo. PESQUISE na "Biblioteca NetApp" para as versões mais atuais.

ONTAP

- "Guia de práticas recomendadas e implementação de NFS"
- "Guia de administração DE SAN" (Para iSCSI)
- "Configuração iSCSI Express para RHEL"

Software Element

- "Configurando o SolidFire para Linux"

NetApp HCI

- "Pré-requisitos de implantação do NetApp HCI"
- "Acesse o mecanismo de implantação do NetApp"

Informações sobre as melhores práticas de aplicação

- "Melhores práticas para MySQL no ONTAP"
- "Melhores práticas para MySQL no SolidFire"
- "NetApp SolidFire e Cassandra"
- "Práticas recomendadas da Oracle no SolidFire"
- "Melhores práticas do PostgreSQL no SolidFire"

Nem todos os aplicativos têm diretrizes específicas, é importante trabalhar com sua equipe do NetApp e usar o "Biblioteca NetApp" para encontrar a documentação mais atualizada.

Integre o Astra Trident

Para integrar o Astra Trident, os seguintes elementos de design e arquitetura exigem integração: Seleção e implantação de drivers, design de classe de storage, design de pool virtual, impacto da reivindicação de volume persistente (PVC) no provisionamento de storage, operações de volume e implantação de serviços OpenShift usando o Astra Trident.

Seleção e implantação do driver

Selecione e implante um driver de back-end para seu sistema de storage.

Drivers de back-end do ONTAP

Os drivers de back-end do ONTAP são diferenciados pelo protocolo usado e pelo modo como os volumes são provisionados no sistema de storage. Portanto, tenha cuidado ao decidir qual driver implantar.

Em um nível mais alto, se seu aplicativo tiver componentes que precisam de armazenamento compartilhado (vários pods acessando o mesmo PVC), os drivers baseados em nas seriam a escolha padrão, enquanto os drivers iSCSI baseados em bloco atendem às necessidades de armazenamento não compartilhado. Escolha o protocolo com base nos requisitos da aplicação e no nível de conforto das equipes de armazenamento e infraestrutura. De um modo geral, há pouca diferença entre eles para a maioria dos aplicativos, portanto, muitas vezes a decisão é baseada na necessidade ou não de armazenamento compartilhado (onde mais de um pod precisará de acesso simultâneo).

Os drivers de back-end ONTAP disponíveis são:

- `ontap-nas`: Cada PV provisionado é um Flexvolume ONTAP completo.
- `ontap-nas-economy`: Cada PV provisionado é uma qtree, com um número configurável de qtrees por Flexvolume (o padrão é 200).
- `ontap-nas-flexgroup`: Cada PV provisionado como um ONTAP FlexGroup completo e todos os agregados atribuídos a um SVM são usados.
- `ontap-san`: Cada PV provisionado é um LUN dentro de seu próprio Flexvolume.
- `ontap-san-economy`: Cada PV provisionado é um LUN, com um número configurável de LUNs por Flexvolume (o padrão é 100).

A escolha entre os três drivers nas tem algumas ramificações para os recursos, que são disponibilizados para o aplicativo.

Observe que, nas tabelas abaixo, nem todos os recursos são expostos pelo Astra Trident. Alguns devem ser aplicados pelo administrador de armazenamento após o provisionamento, se essa funcionalidade for desejada. As notas de rodapé sobrescritas distinguem a funcionalidade por recurso e driver.

Drivers nas ONTAP	Instantâneos	Clones	Políticas de exportação dinâmicas	Ligação múltipla	QoS	Redimensionar	Replicação
`ontap-nas`	Sim	Sim	Nota de rodapé:5[]	Sim	Nota de rodapé:1[]	Sim	Nota de rodapé:1[]
`ontap-nas-economy`	Nota de rodapé:3[]	Nota de rodapé:3[]	Nota de rodapé:5[]	Sim	Nota de rodapé:3[]	Sim	Nota de rodapé:3[]
`ontap-nas-flexgroup`	Nota de rodapé:1[]	Não	Nota de rodapé:5[]	Sim	Nota de rodapé:1[]	Sim	Nota de rodapé:1[]

O Astra Trident oferece 2 drivers SAN para ONTAP, cujas funcionalidades são mostradas abaixo.

Controladores SAN ONTAP	Instantâneos	Clones	Ligação múltipla	CHAP bidirecional	QoS	Redimensionar	Replicação
`ontap-san`	Sim	Sim	Nota de rodapé:4[]	Sim	Nota de rodapé:1[]	Sim	Nota de rodapé:1[]
`ontap-san-economy`	Sim	Sim	Nota de rodapé:4[]	Sim	Nota de rodapé:3[]	Sim	Nota de rodapé:3[]

> Nota de rodapé para as tabelas acima: Yesnote:1[]: Não gerenciado por Astra Trident Yesnote:2[]: Gerenciado por Astra Trident, mas não PV granular Yesnote:3[]: Não gerenciado por Astra Trident e não PV granular Yesnote:4[]: Suportado para volumes em bloco bruto nota de rodapé:5[]: Suportado por Astra Trident

Os recursos que não são granulares PV são aplicados a todo o Flexvolume e todos os PVS (ou seja, qtrees ou LUNs em FlexVols compartilhados) compartilharão um cronograma comum.

Como podemos ver nas tabelas acima, grande parte da funcionalidade entre `ontap-nas` o e `ontap-nas-economy` é a mesma. No entanto, como o `ontap-nas-economy` motorista limita a capacidade de controlar o cronograma em granularidade por PV, isso pode afetar sua recuperação de desastres e Planejamento de backup em particular. Para as equipes de desenvolvimento que desejam utilizar a funcionalidade de clone de PVC no storage ONTAP, isso só é possível ao usar os `ontap-nas` drivers , `ontap-san` ou `ontap-san-economy` .

ⓘ O `solidfire-san` driver também é capaz de clonar PVCs.

Drivers de back-end do Cloud Volumes ONTAP

O Cloud Volumes ONTAP fornece controle de dados junto a recursos de storage de classe empresarial para vários casos de uso, incluindo compartilhamentos de arquivos e storage em nível de bloco, atendendo aos protocolos nas e SAN (NFS, SMB/CIFS e iSCSI). Os drivers compatíveis para o Cloud volume ONTAP são `ontap-nas`, `ontap-nas-economy` `ontap-san` e `ontap-san-economy`. Eles são aplicáveis ao Cloud volume ONTAP para Azure, Cloud volume ONTAP para GCP.

Drivers de back-end do Amazon FSX para ONTAP

O Amazon FSX for NetApp ONTAP permite que você aproveite os recursos, o desempenho e os recursos administrativos do NetApp que você já conhece, ao mesmo tempo em que aproveita a simplicidade, a agilidade, a segurança e a escalabilidade do armazenamento de dados na AWS. O FSX para ONTAP suporta muitos recursos do sistema de arquivos ONTAP e APIs de administração. Os drivers compatíveis para o Cloud volume ONTAP são `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` `ontap-san` e `ontap-san-economy`.

Drivers de back-end NetApp HCI/SolidFire

O `solidfire-san` driver usado com as plataformas NetApp HCI/SolidFire ajuda o administrador a configurar um back-end Element para Trident com base nos limites de QoS. Se você quiser projetar seu back-end para definir os limites de QoS específicos nos volumes provisionados pelo Trident, use o `type` parâmetro no arquivo de back-end. O administrador também pode restringir o tamanho do volume que pode ser criado no armazenamento usando o `limitVolumeSize` parâmetro. Atualmente, recursos de armazenamento de elementos, como redimensionamento de volume e replicação de volume, não são suportados pelo

`solidfire-san` driver. Essas operações devem ser feitas manualmente por meio da IU da Web do Element Software.

Controlador SolidFire	Instantâneos	Clones	Ligação múltipla	CHAP	QoS	Redimensionar	Replicação
`solidfire-san`	Sim	Sim	Nota de rodapé:2[]	Sim	Sim	Sim	Nota de rodapé:1[]

> Nota de rodapé: Yes [1]: Não gerenciado por Astra Trident Yes [2]: Suportado para volumes de blocos brutos

Drivers de back-end do Azure NetApp Files

O Astra Trident usa `azure-netapp-files` o driver para gerenciar o "Azure NetApp Files" serviço.

Mais informações sobre esse driver e como configurá-lo podem ser encontradas no "Configuração de back-end do Astra Trident para Azure NetApp Files".

Controlador Azure NetApp Files	Instantâneos	Clones	Ligação múltipla	QoS	Expandir	Replicação
`azure-netapp-files`	Sim	Sim	Sim	Sim	Sim	Nota de rodapé:1[]

> Nota de rodapé: Yes [1]: Não gerenciado pelo Astra Trident

Cloud Volumes Service no driver de back-end do Google Cloud

O Astra Trident usa `gcp-cvs` o driver para se vincular ao Cloud Volumes Service no Google Cloud.

```
`gcp-cvs`O driver usa pools virtuais para abstrair o back-end e permitir que o Astra Trident determine o posicionamento do volume. O administrador define os pools virtuais nos `backend.json` arquivos. As classes de armazenamento usam seletores para identificar pools virtuais por etiqueta.
```

- Se os pools virtuais forem definidos no back-end, o Astra Trident tentará criar um volume nos pools de storage do Google Cloud ao qual esses pools virtuais são limitados.
- Se os pools virtuais não forem definidos no back-end, o Astra Trident selecionará um pool de storage do Google Cloud nos pools de storage disponíveis na região.

Para configurar o back-end do Google Cloud no Astra Trident, é necessário especificar `projectNumber`, `apiRegion` e `apiKey` no arquivo de back-end. Você pode encontrar o número do projeto no console do Google Cloud. A chave da API é retirada do arquivo de chave privada da conta de serviço que você criou ao configurar o acesso à API para o Cloud Volumes Service no Google Cloud.

Para obter detalhes sobre os tipos de serviço e níveis de serviço do Cloud Volumes Service no Google Cloud, "Saiba mais sobre o suporte ao Astra Trident para CVS para GCP"consulte .

Driver do Cloud Volumes Service para Google Cloud	Instantâneos	Clones	Ligação múltipla	QoS	Expandir	Replicação
`gcp-cvs`	Sim	Sim	Sim	Sim	Sim	Disponível apenas no tipo de serviço CVS-Performance.

Notas de replicação

- A replicação não é gerenciada pelo Astra Trident.
- O clone será criado no mesmo pool de storage que o volume de origem.

Design da classe de armazenamento

As classes de armazenamento individuais precisam ser configuradas e aplicadas para criar um objeto Classe de armazenamento Kubernetes. Esta seção discute como projetar uma classe de armazenamento para seu aplicativo.

Utilização específica no back-end

A filtragem pode ser usada dentro de um objeto de classe de armazenamento específico para determinar qual pool de armazenamento ou conjunto de pools devem ser usados com essa classe de armazenamento específica. Três conjuntos de filtros podem ser definidos na Classe de armazenamento: `storagePools`, `additionalStoragePools` E/ou `excludeStoragePools`.

O `storagePools` parâmetro ajuda a restringir o armazenamento ao conjunto de pools que correspondem a quaisquer atributos especificados. O `additionalStoragePools` parâmetro é usado para estender o conjunto de pools que o Astra Trident usará para provisionar junto com o conjunto de pools selecionados pelos atributos e `storagePools` parâmetros. Você pode usar um parâmetro sozinho ou ambos juntos para garantir que o conjunto apropriado de pools de armazenamento esteja selecionado.

O `excludeStoragePools` parâmetro é usado para excluir especificamente o conjunto listado de pools que correspondem aos atributos.

Emular políticas de QoS

Se você quiser criar classes de armazenamento para emular políticas de qualidade de Serviço, crie uma Classe de armazenamento com o `media` atributo como `hdd` ou `ssd`. Com base no `media` atributo mencionado na classe de storage, o Trident selecionará o back-end apropriado que serve `hdd` ou `ssd` agrega para corresponder ao atributo de Mídia e direcionará o provisionamento dos volumes para o agregado específico. Portanto, podemos criar uma classe de armazenamento PREMIUM que teria um conjunto de atributos, `ssd` que `media` poderia ser classificado como a política de QoS PREMIUM. Podemos criar outro PADRÃO de classe de armazenamento que teria o atributo de Mídia definido como "hdd", que poderia ser classificado como a política de QoS PADRÃO. Também podemos usar o atributo "IOPS" na classe de armazenamento para redirecionar o provisionamento para um dispositivo Element que pode ser definido como uma Política de QoS.

Utilize o back-end com base em recursos específicos

As classes de storage podem ser projetadas para direcionar o provisionamento de volume em um back-end específico, no qual recursos como provisionamento fino e espesso, snapshots, clones e criptografia são ativados. Para especificar qual armazenamento usar, crie classes de armazenamento que especifiquem o back-end apropriado com o recurso necessário habilitado.

Pools virtuais

Os pools virtuais estão disponíveis para todos os back-ends do Astra Trident. Você pode definir pools virtuais para qualquer back-end, usando qualquer driver fornecido pelo Astra Trident.

Os pools virtuais permitem que um administrador crie um nível de abstração sobre backends que pode ser referenciado por meio de classes de armazenamento, para maior flexibilidade e colocação eficiente de volumes em backends. Diferentes backends podem ser definidos com a mesma classe de serviço. Além disso, vários pools de storage podem ser criados no mesmo back-end, mas com características diferentes. Quando uma Classe de armazenamento é configurada com um seletor com as etiquetas específicas, o Astra Trident escolhe um back-end que corresponde a todas as etiquetas do seletor para colocar o volume. Se as etiquetas do seletor de classe de storage corresponderem a vários pools de storage, o Astra Trident escolherá um deles para provisionar o volume.

Design de pool virtual

Ao criar um backend, você geralmente pode especificar um conjunto de parâmetros. Era impossível para o administrador criar outro back-end com as mesmas credenciais de armazenamento e com um conjunto diferente de parâmetros. Com a introdução de pools virtuais, esse problema foi aliviado. Pools virtuais é uma abstração de nível introduzida entre o back-end e a classe de armazenamento do Kubernetes para que o administrador possa definir parâmetros junto com rótulos que podem ser referenciados por meio das classes de armazenamento do Kubernetes como um seletor, de uma forma independente de back-end. Os pools virtuais podem ser definidos para todos os back-ends NetApp compatíveis com o Astra Trident. Essa lista inclui o SolidFire/NetApp HCI, o ONTAP, o Cloud Volumes Service no GCP e o Azure NetApp Files.

Ao definir pools virtuais, é recomendável não tentar reorganizar a ordem dos pools virtuais existentes em uma definição de back-end. Também é aconselhável não editar/modificar atributos para um pool virtual existente e definir um novo pool virtual.

Emulando diferentes níveis de serviço/QoS

É possível projetar pools virtuais para emular classes de serviço. Usando a implementação do pool virtual para o Cloud volume Service for Azure NetApp Files, vamos examinar como podemos configurar diferentes classes de serviço. Configure o back-end do Azure NetApp Files com vários rótulos, representando diferentes níveis de desempenho. Defina `servicelevel` Aspect para o nível de desempenho apropriado e adicione outros aspetos necessários em cada rótulo. Agora crie diferentes classes de armazenamento do Kubernetes que mapeariam para diferentes pools virtuais. Usando o `parameters.selector` campo, cada StorageClass chama quais pools virtuais podem ser usados para hospedar um volume.

Atribuir um conjunto específico de aspetos

Vários pools virtuais com um conjunto específico de aspectos podem ser projetados a partir de um único back-end de storage. Para fazer isso, configure o back-end com vários rótulos e defina os aspetos necessários em cada rótulo. Agora crie diferentes classes de armazenamento do Kubernetes usando o `parameters.selector` campo que mapearia para diferentes pools virtuais. Os volumes que são provisionados no back-end terão os aspetos definidos no pool virtual escolhido.

Caraterísticas de PVC que afetam o provisionamento de armazenamento

Alguns parâmetros além da classe de storage solicitada podem afetar o processo de decisão de provisionamento do Astra Trident ao criar uma PVC.

Modo de acesso

Ao solicitar armazenamento através de um PVC, um dos campos obrigatórios é o modo de acesso. O modo desejado pode afetar o back-end selecionado para hospedar a solicitação de armazenamento.

O Astra Trident tentará corresponder ao protocolo de storage usado com o método de acesso especificado de acordo com a matriz a seguir. Isso é independente da plataforma de storage subjacente.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
ISCSI	Sim	Sim	Sim (bloco bruto)
NFS	Sim	Sim	Sim

Uma solicitação de um PVC ReadWriteMany enviado para uma implantação do Trident sem um back-end NFS configurado resultará em nenhum volume sendo provisionado. Por esse motivo, o solicitante deve usar o modo de acesso apropriado para sua aplicação.

Operações de volume

Modificar volumes persistentes

Volumes persistentes são, com duas exceções, objetos imutáveis no Kubernetes. Uma vez criados, a política de recuperação e o tamanho podem ser modificados. No entanto, isso não impede que alguns aspetos do volume sejam modificados fora do Kubernetes. Isso pode ser desejável para personalizar o volume para aplicações específicas, para garantir que a capacidade não seja consumida acidentalmente ou simplesmente mover o volume para um controlador de armazenamento diferente por qualquer motivo.

> Atualmente, os provisionadores in-tree do Kubernetes não são compatíveis com operações de redimensionamento de volume para PVS NFS ou iSCSI. O Astra Trident é compatível com a expansão de volumes NFS e iSCSI.

Os detalhes de ligação do PV não podem ser modificados após a criação.

Criar snapshots de volume sob demanda

O Astra Trident é compatível com a criação de snapshot de volume sob demanda e a criação de PVCs a partir de snapshots usando a estrutura CSI. Os snapshots fornecem um método conveniente de manter cópias pontuais dos dados e têm um ciclo de vida independente do PV de origem no Kubernetes. Esses snapshots podem ser usados para clonar PVCs.

Criar volumes a partir de instantâneos

O Astra Trident também suporta a criação de PersistentVolumes a partir de instantâneos de volume. Para conseguir isso, basta criar um PersistentVolumeClaim e mencionar o `datasource` como o instantâneo necessário a partir do qual o volume precisa ser criado. O Astra Trident manipulará esse PVC criando um volume com os dados presentes no snapshot. Com esse recurso, é possível duplicar dados entre regiões, criar ambientes de teste, substituir um volume de produção danificado ou corrompido em sua totalidade, ou recuperar arquivos e diretórios específicos e transferi-los para outro volume anexado.

Mover volumes no cluster

Os administradores de storage podem mover volumes entre agregados e controladores no cluster ONTAP sem interrupções para o consumidor de storage. Essa operação não afeta o Astra Trident nem o cluster Kubernetes, contanto que o agregado de destino seja aquele ao qual o SVM que o Astra Trident está usando tenha acesso. É importante ressaltar que se o agregado tiver sido adicionado recentemente ao SVM, o back-end precisará ser atualizado readicionando-o ao Astra Trident. Isso fará com que o Astra Trident faça o inventário novamente da SVM para que o novo agregado seja reconhecido.

No entanto, a movimentação de volumes entre back-ends não é compatível automaticamente com o Astra Trident. Isso inclui entre SVMs no mesmo cluster, entre clusters ou em uma plataforma de storage diferente (mesmo que esse sistema de storage seja conetado ao Astra Trident).

Se um volume for copiado para outro local, o recurso de importação de volume poderá ser usado para importar volumes atuais para o Astra Trident.

Expanda volumes

O Astra Trident é compatível com o redimensionamento de PVS NFS e iSCSI. Isso permite que os usuários redimensionem seus volumes diretamente pela camada Kubernetes. A expansão de volume é possível para todas as principais plataformas de storage da NetApp, incluindo backends ONTAP, SolidFire/NetApp HCI e Cloud Volumes Service. Para permitir uma possível expansão posterior, defina `allowVolumeExpansion` como `true` no StorageClass associado ao volume. Sempre que for necessário redimensionar o volume persistente, edite a `spec.resources.requests.storage` anotação na reclamação volume persistente para o tamanho de volume pretendido. O Trident cuidará automaticamente do redimensionamento do volume no cluster de armazenamento.

Importar um volume existente para o Kubernetes

A importação de volume permite importar um volume de storage existente para um ambiente Kubernetes. Atualmente, isso é suportado pelos `ontap-nas` drivers , `ontap-nas-flexgroup`, `solidfire-san`, `azure-netapp-files` e `gcp-cvs` . Esse recurso é útil ao portar um aplicativo existente para o Kubernetes ou durante cenários de recuperação de desastres.

Ao usar o ONTAP e `solidfire-san` os drivers, use o comando `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` para importar um volume existente para o Kubernetes para ser gerenciado pelo Astra Trident. O arquivo de PVC YAML ou JSON usado no comando de volume de importação aponta para uma classe de storage que identifica o Astra Trident como o provisionador. Ao usar um back-end NetApp HCI/SolidFire, certifique-se de que os nomes de volume sejam exclusivos. Se os nomes de volume forem duplicados, clone o volume para um nome exclusivo para que o recurso de importação de volume possa distinguir entre eles.

Se `azure-netapp-files` o driver ou `gcp-cvs` for usado, use o comando `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` para importar o volume para o Kubernetes para ser gerenciado pelo Astra Trident. Isso garante uma referência de volume única.

Quando o comando acima é executado, o Astra Trident encontrará o volume no back-end e lê seu tamanho. Ele irá adicionar automaticamente (e substituir, se necessário) o tamanho de volume do PVC configurado. Em seguida, o Astra Trident cria o novo PV e o Kubernetes liga o PVC ao PV.

Se um recipiente fosse implantado de modo que fosse necessário o PVC importado específico, ele permaneceria em um estado pendente até que o par PVC/PV seja vinculado através do processo de importação de volume. Depois que o par de PVC / PV são ligados, o recipiente deve surgir, desde que não haja outros problemas.

Implantar serviços OpenShift

Os serviços de cluster de valor agregado do OpenShift fornecem funcionalidade importante aos administradores de cluster e aos aplicativos que estão sendo hospedados. O storage que esses serviços usam pode ser provisionado usando os recursos do nó local. No entanto, isso geralmente limita a capacidade, o desempenho, a capacidade de recuperação e a sustentabilidade do serviço. Ao aproveitar um storage array empresarial para fornecer capacidade a esses serviços, é possível melhorar significativamente o serviço. No entanto, como em todas as aplicações, o OpenShift e os administradores de storage devem trabalhar em conjunto para determinar as melhores opções para cada um. A documentação da Red Hat deve ser muito utilizada para determinar os requisitos e garantir que as necessidades de dimensionamento e desempenho sejam atendidas.

Serviço de registo

A implantação e o gerenciamento do armazenamento para o Registro foram documentados "NetApp.io" "blog"no .

Serviço de registo

Assim como outros serviços OpenShift, o serviço de log é implantado usando o Ansible com parâmetros de configuração fornecidos pelo arquivo de inventário, também conhecido como hosts, fornecidos ao manual de estratégia. Há dois métodos de instalação que serão abordados: Implantação de logs durante a instalação inicial do OpenShift e implantação de logs após a instalação do OpenShift.

> A partir do Red Hat OpenShift versão 3,9, a documentação oficial recomenda contra o NFS para o serviço de log devido a preocupações com a corrupção de dados. Isso é baseado no teste da Red Hat de seus produtos. O servidor NFS do ONTAP não tem esses problemas e pode facilmente fazer backup de uma implantação de log. Em última análise, a escolha do protocolo para o serviço de Registro é sua, apenas saiba que ambos funcionarão muito bem ao usar plataformas NetApp e não há motivo para evitar o NFS se essa for sua preferência.

Se você optar por usar o NFS com o serviço de log, precisará definir a variável Ansible `openshift_enable_unsupported_configurations` para `true` evitar que o instalador falhe.

Comece agora

O serviço de log pode, opcionalmente, ser implantado tanto para aplicativos quanto para as operações principais do próprio cluster OpenShift. Se você optar por implantar o Registro de operações, especificando a variável `openshift_logging_use_ops` como `true`, duas instâncias do serviço serão criadas. As variáveis que controlam a instância de log para operações contêm "OPS" nelas, enquanto a instância para aplicativos não.

A configuração das variáveis do Ansible de acordo com o método de implantação é importante para garantir que o storage correto seja utilizado pelos serviços subjacentes. Vejamos as opções para cada um dos métodos de implantação.

> As tabelas abaixo contêm apenas as variáveis relevantes para a configuração de armazenamento, uma vez que se refere ao serviço de registo. Você pode encontrar outras opções nas "Documentação de Registro do RedHat OpenShift" quais devem ser revisadas, configuradas e usadas de acordo com sua implantação.

As variáveis na tabela abaixo resultarão no manual do Ansible criando um PV e PVC para o serviço de Registro usando os detalhes fornecidos. Esse método é significativamente menos flexível do que usar o

manual de instalação de componentes após a instalação do OpenShift, no entanto, se você tiver volumes existentes disponíveis, é uma opção.

Variável	Detalhes
`openshift_logging_storage_kind`	Defina como `nfs` para que o instalador crie um NFS PV para o serviço de log.
`openshift_logging_storage_host`	O nome do host ou endereço IP do host NFS. Isso deve ser definido para o LIF de dados da sua máquina virtual.
`openshift_logging_storage_nfs_directory`	O caminho de montagem para a exportação NFS. Por exemplo, se o volume for juntado como `/openshift_logging`, você usaria esse caminho para essa variável.
`openshift_logging_storage_volume_name`	O nome, por exemplo `pv_ose_logs`, do PV a criar.
`openshift_logging_storage_volume_size`	O tamanho da exportação NFS, por `100Gi` exemplo .

Se o cluster do OpenShift já estiver em execução e, portanto, o Trident tiver sido implantado e configurado, o instalador poderá usar o provisionamento dinâmico para criar os volumes. As variáveis a seguir precisarão ser configuradas.

Variável	Detalhes
`openshift_logging_es_pvc_dynamic`	Defina como verdadeiro para usar volumes provisionados dinamicamente.
`openshift_logging_es_pvc_storage_class_name`	O nome da classe de armazenamento que será usado no PVC.
`openshift_logging_es_pvc_size`	O tamanho do volume solicitado no PVC.
`openshift_logging_es_pvc_prefix`	Um prefixo para os PVCs usados pelo serviço de Registro.
`openshift_logging_es_ops_pvc_dynamic`	Defina como `true` para usar volumes provisionados dinamicamente para a instância de log de operações.
`openshift_logging_es_ops_pvc_storage_class_name`	O nome da classe de armazenamento para a instância de log de operações.
`openshift_logging_es_ops_pvc_size`	O tamanho da solicitação de volume para a instância de operações.
`openshift_logging_es_ops_pvc_prefix`	Um prefixo para os PVCs de instância de OPS.

Implantar a pilha de logs

Se você estiver implantando o log como parte do processo inicial de instalação do OpenShift, então você só precisará seguir o processo de implantação padrão. O Ansible configurará e implantará os serviços necessários e os objetos OpenShift para que o serviço fique disponível assim que o Ansible for concluído.

No entanto, se você estiver implantando após a instalação inicial, o manual de estratégia de componentes precisará ser usado pelo Ansible. Este processo pode mudar ligeiramente com versões diferentes do OpenShift, portanto, certifique-se de ler e seguir "Documentação do RedHat OpenShift Container Platform 3,11" para a sua versão.

Serviço de métricas

O serviço de métricas fornece informações valiosas ao administrador sobre o status, a utilização de recursos e a disponibilidade do cluster OpenShift. Também é necessário para a funcionalidade de escala automática de pods e muitas organizações usam dados do serviço de métricas para seus aplicativos de cobrança e/ou exibição.

Assim como no serviço de log e no OpenShift como um todo, o Ansible é usado para implantar o serviço de métricas. Além disso, tal como o serviço de registo, o serviço de métricas pode ser implementado durante uma configuração inicial do cluster ou após a sua operação utilizando o método de instalação do componente. As tabelas a seguir contêm as variáveis que são importantes ao configurar o armazenamento persistente para o serviço de métricas.

As tabelas abaixo contêm apenas as variáveis que são relevantes para a configuração de armazenamento, já que se refere ao serviço de métricas. Há muitas outras opções encontradas na documentação que devem ser revisadas, configuradas e usadas de acordo com sua implantação.

Variável	Detalhes
`openshift_metrics_storage_kind`	Defina como `nfs` para que o instalador crie um NFS PV para o serviço de log.
`openshift_metrics_storage_host`	O nome do host ou endereço IP do host NFS. Isso deve ser definido como o LIF de dados para o SVM.
`openshift_metrics_storage_nfs_directory`	O caminho de montagem para a exportação NFS. Por exemplo, se o volume for juntado como `/openshift_metrics`, você usaria esse caminho para essa variável.
`openshift_metrics_storage_volume_name`	O nome, por exemplo `pv_ose_metrics`, do PV a criar.
`openshift_metrics_storage_volume_size`	O tamanho da exportação NFS, por `100Gi` exemplo .

Se o cluster do OpenShift já estiver em execução e, portanto, o Trident tiver sido implantado e configurado, o instalador poderá usar o provisionamento dinâmico para criar os volumes. As variáveis a seguir precisarão ser configuradas.

Variável	Detalhes
`openshift_metrics_cassandra_pvc_prefix`	Um prefixo a ser usado para as PVCs de métricas.
`openshift_metrics_cassandra_pvc_size`	O tamanho dos volumes a solicitar.
`openshift_metrics_cassandra_storage_type`	O tipo de storage a ser usado para métricas, isso precisa ser definido como dinâmico para que o Ansible crie PVCs com a classe de storage apropriada.
`openshift_metrics_cassanda_pvc_storage_class_name`	O nome da classe de armazenamento a utilizar.

Implantar o serviço de métricas

Com as variáveis apropriadas do Ansible definidas no arquivo de hosts/inventário, implante o serviço com o Ansible. Se você estiver implantando no horário de instalação do OpenShift, o PV será criado e usado automaticamente. Se você estiver implantando usando os playbooks de componentes, após a instalação do OpenShift, o Ansible criará todos os PVCs necessários e, depois que o Astra Trident provisionou o storage para eles, implantará o serviço.

As variáveis acima, e o processo de implantação, podem mudar com cada versão do OpenShift. Certifique-se de rever e seguir "Guia de implantação do OpenShift da RedHat" a sua versão para que ela seja configurada para o seu ambiente.

Proteção de dados e recuperação de desastres

Saiba mais sobre as opções de proteção e recuperação para o Astra Trident e volumes criados com o Astra Trident. Você deve ter uma estratégia de proteção e recuperação de dados para cada aplicação com um requisito de persistência.

Replicação e recuperação do Astra Trident

Você pode criar um backup para restaurar o Astra Trident em caso de desastre.

Replicação do Astra Trident

O Astra Trident usa CRDs do Kubernetes para armazenar e gerenciar seu próprio estado e o cluster do Kubernetes etcd para armazenar seus metadados.

Passos

1. Faça backup do cluster do Kubernetes etcd usando "Kubernetes: Fazer backup de um cluster etcd"o .
2. Coloque os artefatos de backup em um FlexVol.

> Recomendamos que você proteja o SVM em que o FlexVol reside com uma relação do SnapMirror com outro SVM.

Recuperação do Astra Trident

Usando as CRDs do Kubernetes e o snapshot etcd do cluster do Kubernetes, é possível recuperar o Astra Trident.

Passos

1. No SVM de destino, monte o volume que contém os arquivos de dados e certificados do Kubernetes no host, que será configurado como um nó mestre.
2. Copie todos os certificados necessários referentes ao cluster do Kubernetes `/etc/kubernetes/pki` e os arquivos de membros do etcd em `/var/lib/etcd`.
3. Restaure o cluster do Kubernetes a partir do backup etcd usando "Kubernetes: Restaurando um cluster etcd"o .
4. Execute `kubectl get crd` para verificar se todos os recursos personalizados do Trident foram criados e recuperar os objetos Trident para verificar se todos os dados estão disponíveis.

Replicação e recuperação da SVM

No entanto, o Astra Trident não pode configurar relações de replicação. No entanto, o administrador de storage pode usar "ONTAP SnapMirror" para replicar uma SVM.

Em caso de desastre, você pode ativar o SVM de destino do SnapMirror para começar a fornecer dados. Você pode voltar para o primário quando os sistemas são restaurados.

Sobre esta tarefa

Considere o seguinte ao usar o recurso de replicação do SnapMirror SVM:

- Você deve criar um back-end distinto para cada SVM com SVM-DR ativado.
- Configure as classes de storage para selecionar os back-ends replicados somente quando necessário, a fim de evitar ter volumes que não precisam de replicação provisionados nos back-ends compatíveis com SVM-DR.
- Os administradores de aplicativos devem entender o custo e a complexidade adicionais associados à replicação e considerar cuidadosamente seu plano de recuperação antes de iniciar esse processo.

Replicação da SVM

Você pode usar "ONTAP: Replicação do SnapMirror SVM" o para criar a relação de replicação do SVM.

O SnapMirror permite que você defina opções para controlar o que replicar. Você precisará saber quais opções você selecionou ao pré-formar Recuperação do SVM com Astra Trident.

- "-identidade-preservar verdadeiro" Replica toda a configuração da SVM.
- "-discard-configs network" Exclui LIFs e configurações de rede relacionadas.
- "-identity-preserve false" replica apenas os volumes e a configuração de segurança.

Recuperação do SVM com Astra Trident

O Astra Trident não detecta automaticamente falhas na SVM. Em caso de desastre, o administrador pode iniciar manualmente o failover de Trident para a nova SVM.

Passos

1. Cancele transferências de SnapMirror agendadas e contínuas, interrompa a relação de replicação, pare o SVM de origem e ative o SVM de destino do SnapMirror.
2. Se você especificou `-identity-preserve false` ou `-discard-config network` ao configurar sua replicação SVM, atualize o `managementLIF` e `dataLIF` no arquivo de definição de back-end do Trident.
3. `storagePrefix`Confirmar está presente no arquivo de definição de back-end do Trident. Este parâmetro não pode ser alterado. Omitir `storagePrefix` fará com que a atualização de backend falhe.
4. Atualize todos os backends necessários para refletir o novo nome SVM de destino usando:

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. Se você especificou `-identity-preserve false` ou `discard-config network`, você deve rejeitar todos os pods de aplicativo.

Se você especificou `-identity-preserve true`, todos os volumes provisionados pelo Astra Trident começam a fornecer dados quando o SVM de destino é ativado.

Replicação de volume e recuperação

No entanto, o Astra Trident não pode configurar relações de replicação do SnapMirror. No entanto, o administrador de storage pode usar "Replicação e recuperação do ONTAP SnapMirror" para replicar volumes criados pelo Astra Trident.

Em seguida, você pode importar os volumes recuperados para o Astra Trident usando "importação de volume tridentctl"o .

A importação não é suportada em `ontap-nas-economy` drivers , `ontap-san-economy`, ou `ontap-flexgroup-economy` .

Proteção de dados do Snapshot

Você pode proteger e restaurar dados usando:

- Uma controladora de snapshot externa e CRDs para criar snapshots de volume do Kubernetes de volumes persistentes (PVS).

  "Instantâneos de volume"

- Snapshots ONTAP para restaurar todo o conteúdo de um volume ou recuperar arquivos individuais ou LUNs.

  "Snapshots ONTAP"

Replicação de aplicações Astra Control Center

Com o Astra Control, você pode replicar alterações de dados e aplicações de um cluster para outro usando as funcionalidades de replicação assíncrona do SnapMirror.

"Astra Control: Replique aplicações para um sistema remoto usando a tecnologia SnapMirror"

Segurança

Segurança

Use as recomendações listadas aqui para garantir a segurança da instalação do seu Astra Trident.

Execute o Astra Trident em seu próprio namespace

É importante impedir que aplicações, administradores de aplicações, usuários e aplicações de gerenciamento acessem as definições de objetos do Astra Trident ou os pods para garantir um storage confiável e bloquear atividades maliciosas em potencial.

Para separar as outras aplicações e usuários do Astra Trident, instale sempre o Astra Trident em seu próprio

namespace Kubernetes (`trident`). A colocação do Astra Trident em seu próprio namespace garante que apenas o pessoal administrativo do Kubernetes tenha acesso ao pod Astra Trident e aos artefatos (como segredos de back-end e CHAP, se aplicável) armazenados nos objetos CRD com namespaces. Você deve garantir que somente os administradores acessem o namespace Astra Trident e, assim, o acesso `tridentctl` à aplicação.

Use a autenticação CHAP com backends ONTAP SAN

O Astra Trident é compatível com autenticação baseada em CHAP para workloads SAN ONTAP (usando os `ontap-san` drivers e `ontap-san-economy`). A NetApp recomenda o uso de CHAP bidirecional com Astra Trident para autenticação entre um host e o back-end de storage.

Para backends ONTAP que usam os drivers de armazenamento SAN, o Astra Trident pode configurar CHAP bidirecional e gerenciar nomes de usuário e segredos do CHAP por meio `tridentctl`do . ""Consulte para entender como o Astra Trident configura o CHAP nos backends do ONTAP.

Use a autenticação CHAP com backends NetApp HCI e SolidFire

O NetApp recomenda a implantação de CHAP bidirecional para garantir a autenticação entre um host e os backends NetApp HCI e SolidFire. O Astra Trident usa um objeto secreto que inclui duas senhas CHAP por locatário. Quando o Astra Trident é instalado, ele gerencia os segredos CHAP e os armazena em um `tridentvolume` objeto CR para o respetivo PV. Quando você cria um PV, o Astra Trident usa os segredos CHAP para iniciar uma sessão iSCSI e se comunicar com o sistema NetApp HCI e SolidFire através do CHAP.

Os volumes criados pelo Astra Trident não estão associados a nenhum grupo de acesso a volume.

Use o Astra Trident com NVE e NAE

O NetApp ONTAP fornece criptografia de dados em repouso para proteger dados confidenciais caso um disco seja roubado, retornado ou reutilizado. Para obter detalhes, "Configurar a visão geral da encriptação de volume do NetApp"consulte .

- Se o NAE estiver ativado no back-end, qualquer volume provisionado no Astra Trident será habilitado para NAE.
- Se o NAE não estiver habilitado no back-end, qualquer volume provisionado no Astra Trident será habilitado para NVE, a menos que você defina o sinalizador de criptografia NVE como `false` na configuração de back-end.

Os volumes criados no Astra Trident em um back-end habilitado para NAE devem ser criptografados com NVE ou NAE.

- Você pode definir o sinalizador de criptografia NVE como `true` na configuração de back-end do Trident para substituir a criptografia NAE e usar uma chave de criptografia específica por volume.
- Definir o sinalizador de criptografia NVE como `false` em um back-end habilitado para NAE criará um volume habilitado para NAE. Não é possível desativar a criptografia NAE definindo o sinalizador de criptografia NVE como `false`.

- Você pode criar manualmente um volume NVE no Astra Trident definindo explicitamente o sinalizador de criptografia NVE como `true`.

Para obter mais informações sobre opções de configuração de back-end, consulte:

- "Opções de configuração de SAN ONTAP"
- "Opções de configuração do ONTAP nas"

Configuração de chave unificada do Linux (LUKS)

Você pode ativar o LUKS (configuração de chave unificada do Linux) para criptografar volumes DE ECONOMIA SAN ONTAP e SAN ONTAP no Astra Trident. O Astra Trident é compatível com rotação de frase-passe e expansão de volume para volumes criptografados por LUKS.

No Astra Trident, os volumes criptografados por LUKS usam a cifra e o modo aes-xts-plain64, conforme recomendado "NIST"pelo .

Antes de começar

- Os nós de trabalho devem ter o cryptsetup 2,1 ou superior (mas inferior a 3,0) instalado. Para obter mais informações, visite "Gitlab: Cryptsetup".
- Por motivos de desempenho, recomendamos que os nós de trabalho suportem Advanced Encryption Standard New Instructions (AES-NI). Para verificar o suporte ao AES-NI, execute o seguinte comando:

  ```
  grep "aes" /proc/cpuinfo
  ```

  Se nada for devolvido, o processador não suporta AES-NI. Para obter mais informações sobre o AES-NI, visite: "Intel: Advanced Encryption Standard Instructions (AES-NI)".

Ativar encriptação LUKS

Você pode ativar a criptografia por volume no lado do host usando o LUKS (Configuração de chave unificada do Linux) para volumes ECONÔMICOS SAN ONTAP e SAN ONTAP.

Passos

1. Defina atributos de criptografia LUKS na configuração de back-end. Para obter mais informações sobre opções de configuração de back-end para SAN ONTAP, "Opções de configuração de SAN ONTAP"consulte .

```
"storage": [
    {
        "labels":{"luks": "true"},
        "zone":"us_east_1a",
        "defaults": {
            "luksEncryption": "true"
        }
    },
    {
        "labels":{"luks": "false"},
        "zone":"us_east_1a",
        "defaults": {
            "luksEncryption": "false"
        }
    },
]
```

2. Use `parameters.selector` para definir os pools de armazenamento usando a criptografia LUKS. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. Crie um segredo que contenha a frase-passe LUKS. Por exemplo:

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

Limitações

Os volumes criptografados com LUKS não podem aproveitar a deduplicação e a compactação do ONTAP.

Configuração de back-end para importação de volumes LUKS

Para importar um volume LUKS, você deve definir `luksEncryption` como(`true` no back-end. A `luksEncryption` opção informa ao Astra Trident se o volume é compatível com LUKS (`true`) ou não com LUKS (`false`), conforme mostrado no exemplo a seguir.

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

Rode uma frase-passe LUKS

Pode rodar a frase-passe LUKS e confirmar a rotação.

> Não se esqueça de uma frase-passe até ter verificado que ela não é mais referenciada por qualquer volume, instantâneo ou segredo. Se uma frase-passe referenciada for perdida, talvez você não consiga montar o volume e os dados permanecerão criptografados e inacessíveis.

Sobre esta tarefa

A rotação da frase-passe LUKS ocorre quando um pod que monta o volume é criado após uma nova frase-passe LUKS ser especificada. Quando um novo pod é criado, o Astra Trident compara a frase-passe LUKS no volume com a frase-passe ativa em segredo.

- Se a frase-passe no volume não corresponder à frase-passe ativa no segredo, ocorre rotação.
- Se a frase-passe no volume corresponder à frase-passe ativa no segredo, o `previous-luks-passphrase` parâmetro é ignorado.

Passos

1. Adicione os `node-publish-secret-name` parâmetros e `node-publish-secret-namespace` StorageClass. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. Identificar senhas existentes no volume ou instantâneo.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["A"]
```

3. Atualize o segredo LUKS para o volume para especificar as senhas novas e anteriores. Certifique-se `previous-luke-passphrase-name` e `previous-luks-passphrase` faça a correspondência da frase-passe anterior.

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. Crie um novo pod de montagem do volume. Isto é necessário para iniciar a rotação.
5. Verifique se a senha foi girada.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

Resultados

A frase-passe foi girada quando apenas a nova frase-passe é retornada no volume e no instantâneo.

Se duas senhas forem retornadas, por `luksPassphraseNames: ["B", "A"]` exemplo, a rotação estará incompleta. Você pode acionar um novo pod para tentar completar a rotação.

Ative a expansão de volume

Você pode ativar a expansão de volume em um volume criptografado com LUKS.

Passos

1. Ative a `CSINodeExpandSecret` porta de recurso (beta 1,25 ou mais). "Kubernetes 1,25: Use segredos para a expansão orientada por nós de volumes CSI"Consulte para obter detalhes.
2. Adicione os `node-expand-secret-name` parâmetros e `node-expand-secret-namespace` StorageClass. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

Resultados

Quando você inicia a expansão de armazenamento on-line, o kubelet passa as credenciais apropriadas para o

driver.

Conhecimento e apoio

Perguntas frequentes

Encontre respostas para as perguntas mais frequentes sobre a instalação, configuração, atualização e solução de problemas do Astra Trident.

Questões gerais

Com que frequência o Astra Trident é lançado?

A partir do lançamento de 24,02, o Astra Trident é lançado a cada quatro meses: Fevereiro, junho e outubro.

O Astra Trident é compatível com todos os recursos lançados em uma versão específica do Kubernetes?

O Astra Trident geralmente não é compatível com recursos alfa do Kubernetes. O Trident pode oferecer suporte a recursos beta nas duas versões do Trident que seguem a versão beta do Kubernetes.

O Astra Trident tem alguma dependência de outros produtos NetApp para seu funcionamento?

O Astra Trident não tem dependências em outros produtos de software NetApp e funciona como uma aplicação autônoma. No entanto, você deve ter um dispositivo de storage de back-end do NetApp.

Como posso obter detalhes completos da configuração do Astra Trident?

Use o `tridentctl get` comando para obter mais informações sobre sua configuração do Astra Trident.

Posso obter métricas sobre como o storage é provisionado pelo Astra Trident?

Sim. Endpoints Prometheus que podem ser usados para coletar informações sobre a operação do Astra Trident, como o número de backends gerenciados, o número de volumes provisionados, bytes consumidos e assim por diante. Você também pode usar "Cloud Insights" para monitoramento e análise.

A experiência do usuário muda ao usar o Astra Trident como um supervisor CSI?

Não há alterações no que diz respeito à experiência do usuário e às funcionalidades. O nome do provisionador usado é `csi.trident.netapp.io`. Esse método de instalação do Astra Trident é recomendado se você quiser usar todos os novos recursos fornecidos pelas versões atuais e futuras.

Instalar e usar o Astra Trident em um cluster Kubernetes

O Astra Trident oferece suporte a uma instalação off-line a partir de um Registro privado?

Sim, o Astra Trident pode ser instalado offline. "Saiba mais sobre a instalação do Astra Trident"Consulte a .

Posso instalar o Astra Trident be remotamente?

Sim. O Astra Trident 18,10 e posterior são compatíveis com a funcionalidade de instalação remota de qualquer máquina que tenha `kubectl` acesso ao cluster. Depois `kubectl` que o acesso for verificado (por exemplo, inicie um `kubectl get nodes` comando da máquina remota para verificar), siga as instruções de instalação.

Posso configurar a alta disponibilidade com o Astra Trident?

O Astra Trident é instalado como uma implantação do Kubernetes (ReplicaSet) com uma instância, e por isso tem HA incorporada. Você não deve aumentar o número de réplicas na implantação. Se o nó em que o Astra Trident está instalado for perdido ou o pod estiver inacessível, o Kubernetes reimplanta automaticamente o pod em um nó íntegro no cluster. O Astra Trident é apenas um plano de controle. Portanto, os pods montados atualmente não são afetados se o Astra Trident for reimplantado.

O Astra Trident precisa de acesso ao namespace do sistema kube?

O Astra Trident lê o servidor de API Kubernetes para determinar quando as aplicações solicitam novos PVCs de modo que a TI precisa de acesso ao sistema kube.

Quais são as funções e Privileges usadas pelo Astra Trident?

O instalador do Trident cria um ClusterRole Kubernetes, que tem acesso específico aos recursos Persistentvolume, PersistentVolumeClaim, StorageClass e segredo do cluster do Kubernetes. "Personalize a instalação do tridentctl"Consulte a .

Posso gerar localmente os arquivos de manifesto exatos que o Astra Trident usa para instalação?

Você pode gerar e modificar localmente os arquivos de manifesto exatos que o Astra Trident usa para instalação, se necessário. "Personalize a instalação do tridentctl"Consulte a .

Posso compartilhar o mesmo SVM de back-end do ONTAP em duas instâncias separadas do Astra Trident em dois clusters Kubernetes separados?

Embora não seja aconselhado, você pode usar o mesmo SVM de back-end em duas instâncias do Astra Trident. Especifique um nome de volume exclusivo para cada instância durante a instalação e/ou especifique um parâmetro exclusivo `StoragePrefix` no `setup/backend.json` arquivo. Isso serve para garantir que o mesmo FlexVol não seja usado para ambas as instâncias.

É possível instalar o Astra Trident sob o ContainerLinux (antigo CoreOS)?

O Astra Trident é simplesmente um pod de Kubernetes e pode ser instalado onde quer que o Kubernetes esteja em execução.

Posso usar o Astra Trident com NetApp Cloud Volumes ONTAP?

Sim, o Astra Trident é compatível com AWS, Google Cloud e Azure.

O Astra Trident funciona com o Cloud volumes Services?

Sim, o Astra Trident é compatível com o serviço Azure NetApp Files no Azure e com o Cloud Volumes Service no GCP.

Solução de problemas e suporte

O NetApp é compatível com Astra Trident?

Embora o Astra Trident seja de código aberto e fornecido gratuitamente, o NetApp oferece suporte total desde que o back-end do NetApp seja compatível.

Como faço para levantar um caso de suporte?

Para levantar um caso de suporte, execute um dos seguintes procedimentos:

1. Entre em Contato com seu gerente de conta de suporte e obtenha ajuda para levantar um ticket.
2. Envie um caso de suporte entrando em Contato "Suporte à NetApp"com .

Como gerar um pacote de log de suporte?

Você pode criar um pacote de suporte executando `tridentctl logs -a`o . Além dos logs capturados no pacote, capture o log do kubelet para diagnosticar os problemas de montagem no lado do Kubernetes. As instruções para obter o log do kubelet variam de acordo com a forma como o Kubernetes é instalado.

O que devo fazer se for necessário enviar uma solicitação para um novo recurso?

Crie um problema "Astra Trident GitHub" e mencione **RFE** no assunto e na descrição do problema.

Onde posso levantar um defeito?

Crie um problema no "Astra Trident GitHub". Certifique-se de incluir todas as informações e logs necessários relativos ao problema.

O que acontece se eu tiver uma pergunta rápida sobre o Astra Trident sobre a qual preciso de esclarecimentos? Existe uma comunidade ou um fórum?

Se você tiver dúvidas, problemas ou solicitações, entre em Contato conosco através do nosso Astra "Canal discord" ou GitHub.

A senha do meu sistema de storage mudou e o Astra Trident não funciona mais. Como faço para recuperar?

Atualize a senha do backend com `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`o . Substitua `myBackend` no exemplo pelo nome do backend e `/path/to_new_backend.json` pelo caminho para o arquivo correto `backend.json`.

O Astra Trident não encontra meu nó Kubernetes. Como faço para corrigir isso?

Há dois cenários prováveis pelos quais o Astra Trident não consegue encontrar um nó Kubernetes. Pode ser devido a um problema de rede no Kubernetes ou a um problema de DNS. O daemonset do nó do Trident que é executado em cada nó do Kubernetes deve ser capaz de se comunicar com o controlador Trident para Registrar o nó no Trident. Se as alterações de rede ocorrerem após a instalação do Astra Trident, você encontrará esse problema apenas com novos nós Kubernetes adicionados ao cluster.

Se o pod Trident for destruído, eu perderei os dados?

Os dados não serão perdidos se o pod Trident for destruído. Os metadados do Trident são armazenados em objetos CRD. Todos os PVS que foram provisionados pelo Trident funcionarão normalmente.

Atualizar o Astra Trident

Posso atualizar de uma versão mais antiga diretamente para uma versão mais recente (ignorando algumas versões)?

A NetApp oferece suporte à atualização do Astra Trident de um grande lançamento para o próximo grande lançamento imediato. Você pode atualizar da versão 18.xx para 19.xx, 19.xx para 20.xx, e assim por diante. Você deve testar a atualização em um laboratório antes da implantação da produção.

É possível fazer o downgrade do Trident para uma versão anterior?

Se você precisar de uma correção para bugs observados após uma atualização, problemas de dependência ou uma atualização mal sucedida ou incompleta, você deve "Desinstale o Astra Trident"reinstalar a versão anterior usando as instruções específicas para essa versão. Esta é a única maneira recomendada de fazer o downgrade para uma versão anterior.

Gerenciar backends e volumes

Preciso definir o gerenciamento e LIFs de dados em um arquivo de definição de back-end do ONTAP?

O LIF de gestão é obrigatório. O LIF de dados varia:

- ONTAP SAN: Não especifique para iSCSI. O Astra Trident usa "Mapa de LUN seletivo da ONTAP" para descobrir os LIFs iSCI necessários para estabelecer uma sessão de vários caminhos. Um aviso é gerado se `dataLIF` for definido explicitamente. "Exemplos e opções de configuração de SAN ONTAP"Consulte para obter detalhes.
- ONTAP nas: Recomendamos especificar `dataLIF`. Se não for fornecido, o Astra Trident obtém LIFs de dados do SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) a ser usado para as operações de montagem NFS, permitindo que você crie um DNS de round-robin para balanceamento de carga em vários LIFs de dados. "Exemplos e opções de configuração do ONTAP nas"Consulte para obter detalhes

O Astra Trident pode configurar o CHAP para backends ONTAP?

Sim. O Astra Trident é compatível com CHAP bidirecional para backends ONTAP. Isso requer configuração `useCHAP=true` em sua configuração de back-end.

Como faço para gerenciar políticas de exportação com o Astra Trident?

O Astra Trident pode criar e gerenciar políticas de exportação dinamicamente a partir da versão 20,04. Isso permite que o administrador de storage forneça um ou mais blocos CIDR em sua configuração de back-end e que o Trident adicione IPs de nós que se enquadram nesses intervalos a uma política de exportação criada por ele. Dessa forma, o Astra Trident gerencia automaticamente a adição e exclusão de regras para nós com IPs nos CIDR fornecidos.

Os endereços IPv6 podem ser usados para os LIFs de gerenciamento e dados?

O Astra Trident é compatível com a definição de endereços IPv6 para:

- `managementLIF` E `dataLIF` para backends ONTAP nas.
- `managementLIF` Para backends ONTAP SAN. Não é possível especificar `dataLIF` em um back-end de SAN ONTAP.

O Astra Trident deve ser instalado usando o `--use-ipv6` sinalizador (`tridentctl`para instalação), `IPv6` (para o operador Trident) ou `tridentTPv6` (para instalação Helm) para que ele funcione acima de

IPv6.

É possível atualizar o LIF de gerenciamento no back-end?

Sim, é possível atualizar o backend Management LIF usando o `tridentctl update backend` comando.

É possível atualizar o Data LIF no backend?

Você pode atualizar o Data LIF em `ontap-nas` e `ontap-nas-economy` somente.

Posso criar vários back-ends no Astra Trident para Kubernetes?

O Astra Trident pode dar suporte a muitos backends simultaneamente, seja com o mesmo driver ou com drivers diferentes.

Como o Astra Trident armazena credenciais de back-end?

O Astra Trident armazena as credenciais de back-end como segredos do Kubernetes.

Como o Astra Trident seleciona um back-end específico?

Se os atributos de back-end não puderem ser usados para selecionar automaticamente os pools corretos para uma classe, os `storagePools` parâmetros e `additionalStoragePools` serão usados para selecionar um conjunto específico de pools.

Como posso garantir que o Astra Trident não provisione de um back-end específico?

O `excludeStoragePools` parâmetro é usado para filtrar o conjunto de pools que o Astra Trident usará para provisionar e removerá todos os pools correspondentes.

Se houver vários backends do mesmo tipo, como o Astra Trident seleciona qual back-end usar?

Se houver vários backends configurados do mesmo tipo, o Astra Trident seleciona o back-end apropriado com base nos parâmetros presentes no `StorageClass` e `PersistentVolumeClaim`no . Por exemplo, se houver vários backends de driver ONTAP-nas, o Astra Trident tentará corresponder parâmetros no `StorageClass` e `PersistentVolumeClaim` combinou e corresponder a um back-end que possa atender aos requisitos listados em `StorageClass` e `PersistentVolumeClaim`. Se houver vários backends que correspondam à solicitação, o Astra Trident seleciona um deles aleatoriamente.

O Astra Trident é compatível com CHAP bidirecional com Element/SolidFire?

Sim.

Como o Astra Trident implanta Qtrees em um volume ONTAP? Quantos Qtrees podem ser implantados em um único volume?

`ontap-nas-economy`O driver cria até 200 Qtrees no mesmo FlexVol (configurável entre 50 e 300), 100.000 Qtrees por nó de cluster e 2,4M por cluster. Quando você insere um novo `PersistentVolumeClaim` que é atendido pelo driver de economia, o driver procura ver se já existe um FlexVol que pode atender o novo Qtree. Se o FlexVol não existir que possa servir o Qtree, um novo FlexVol será criado.

Como posso definir permissões Unix para volumes provisionados no ONTAP nas?

Você pode definir permissões Unix no volume provisionado pelo Astra Trident definindo um parâmetro no arquivo de definição de back-end.

Como posso configurar um conjunto explícito de opções de montagem ONTAP NFS enquanto provisiono um volume?

Por padrão, o Astra Trident não define as opções de montagem como nenhum valor com o Kubernetes. Para especificar as opções de montagem na classe de armazenamento do Kubernetes, siga o exemplo fornecido "aqui".

Como faço para definir os volumes provisionados para uma política de exportação específica?

Para permitir que os hosts apropriados acessem um volume, use o `exportPolicy` parâmetro configurado no arquivo de definição de back-end.

Como definir a criptografia de volumes por meio do Astra Trident com ONTAP?

Você pode definir a criptografia no volume provisionado pelo Trident usando o parâmetro de criptografia no arquivo de definição de back-end. Para obter mais informações, consulte: "Como o Astra Trident funciona com NVE e NAE"

Qual é a melhor maneira de implementar QoS para ONTAP por meio do Astra Trident?

`StorageClasses`Use para implementar QoS para ONTAP.

Como especificar o provisionamento thin ou thick por meio do Astra Trident?

Os drivers ONTAP oferecem suporte ao provisionamento thin ou thick. Os drivers do ONTAP são padrão para thin Provisioning. Se o provisionamento espesso for desejado, você deverá configurar o arquivo de definição de back-end ou o `StorageClass`. Se ambos estiverem configurados, `StorageClass` tem precedência. Configure o seguinte para o ONTAP:

1. On `StorageClass`, defina o `provisioningType` atributo como thick (espesso).
2. No arquivo de definição de back-end, ative volumes espessos definindo `backend spaceReserve parameter` como volume.

Como posso garantir que os volumes que estão a ser utilizados não sejam eliminados mesmo que elimine acidentalmente o PVC?

A proteção de PVC é ativada automaticamente no Kubernetes a partir da versão 1,10.

Posso expandir PVCs de NFS criados pelo Astra Trident?

Sim. Você pode expandir um PVC que foi criado pelo Astra Trident. Observe que o volume com crescimento automático é um recurso do ONTAP que não é aplicável ao Trident.

Posso importar um volume enquanto estiver no modo de proteção de dados (DP) da SnapMirror ou offline?

A importação de volume falha se o volume externo estiver no modo DP ou estiver offline. Você recebe a seguinte mensagem de erro:

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

Como a cota de recursos é traduzida para um cluster NetApp?

A cota de recursos de armazenamento do Kubernetes deve funcionar enquanto o armazenamento do NetApp tiver capacidade. Quando o storage do NetApp não consegue atender às configurações de cota do Kubernetes devido à falta de capacidade, o Astra Trident tenta provisionar, mas faz erros.

Posso criar snapshots de volume usando o Astra Trident?

Sim. A criação de snapshots de volume sob demanda e volumes persistentes a partir de snapshots é compatível com o Astra Trident. Para criar PVS a partir de instantâneos, certifique-se de que a `VolumeSnapshotDataSource` porta de recurso foi ativada.

Quais são os drivers compatíveis com snapshots de volume Astra Trident?

A partir de hoje, o suporte a snapshot sob demanda está disponível para o nosso `ontap-nas` `ontap-nas-flexgroup` , `ontap-san`, `ontap-san-economy`, , , `solidfire-san` `gcp-cvs`, e `azure-netapp-files` drivers de back-end.

Como faço para fazer um backup instantâneo de um volume provisionado pelo Astra Trident com ONTAP?

Isso está disponível nos `ontap-nas` drivers , `ontap-san` e `ontap-nas-flexgroup` . Você também pode especificar um `snapshotPolicy` para o `ontap-san-economy` driver no nível FlexVol.

Isso também está disponível `ontap-nas-economy` nos drivers, mas na granularidade de nível FlexVol e não na granularidade de nível de qtree. Para habilitar a capacidade de snapshot volumes provisionados pelo Astra Trident, defina a opção de parâmetro de back-end `snapshotPolicy` para a política de snapshot desejada, conforme definido no back-end do ONTAP. Todos os snapshots feitos pelo controlador de storage não são conhecidos pelo Astra Trident.

Posso definir uma porcentagem de reserva de snapshot para um volume provisionado por meio do Astra Trident?

Sim, você pode reservar uma porcentagem específica de espaço em disco para armazenar as cópias snapshot por meio do Astra Trident definindo `snapshotReserve` o atributo no arquivo de definição de back-end. Se você configurou `snapshotPolicy` e `snapshotReserve` no arquivo de definição de back-end, a porcentagem de reserva de snapshot é definida de acordo com a `snapshotReserve` porcentagem mencionada no arquivo de back-end. Se o `snapshotReserve` número percentual não for mencionado, ONTAP por padrão leva a porcentagem de reserva de snapshot como 5. Se a `snapshotPolicy` opção estiver definida como None (nenhum), a percentagem de reserva de instantâneos é definida como 0.

Posso acessar diretamente o diretório instantâneo do volume e copiar arquivos?

Sim, você pode acessar o diretório instantâneo no volume provisionado pelo Trident definindo o `snapshotDir` parâmetro no arquivo de definição de back-end.

Posso configurar o SnapMirror para volumes com o Astra Trident?

Atualmente, o SnapMirror precisa ser definido externamente usando a CLI ou o OnCommand System Manager do ONTAP.

Como faço para restaurar volumes persistentes para um snapshot específico do ONTAP?

Para restaurar um volume para um instantâneo do ONTAP, execute as seguintes etapas:

1. Quiesce o pod do aplicativo que está usando o volume persistente.
2. Reverter para o snapshot necessário por meio da CLI ou OnCommand System Manager do ONTAP.
3. Reinicie o pod de aplicativos.

O Trident provisiona volumes em SVMs que têm um espelhamento de compartilhamento de carga configurado?

Os espelhos de compartilhamento de carga podem ser criados para volumes raiz de SVMs que fornecem dados por NFS. O ONTAP atualiza automaticamente os espelhos de compartilhamento de carga para volumes criados pelo Trident. Isso pode resultar em atrasos nos volumes de montagem. Quando vários volumes são criados usando o Trident, o provisionamento de um volume depende da atualização do espelhamento de compartilhamento de carga do ONTAP.

Como posso separar o uso da classe de storage para cada cliente/locatário?

O Kubernetes não permite classes de storage em namespaces. No entanto, você pode usar o Kubernetes para limitar o uso de uma classe de armazenamento específica por namespace usando cotas de recursos de armazenamento, que são por namespace. Para negar acesso a um namespace específico a um armazenamento específico, defina a cota de recurso como 0 para essa classe de armazenamento.

Solução de problemas

Use os ponteiros fornecidos aqui para solucionar problemas que você pode encontrar ao instalar e usar o Astra Trident.

Resolução de problemas gerais

- Se o pod Trident não aparecer corretamente (por exemplo, quando o pod Trident está preso `ContainerCreating` na fase com menos de dois contêineres prontos), em execução `kubectl -n trident describe deployment trident` e `kubectl -n trident describe pod trident-**` pode fornecer informações adicionais. A obtenção de logs do kubelet (por exemplo, via `journalctl -xeu kubelet`) também pode ser útil.
- Se não houver informações suficientes nos logs do Trident, você pode tentar ativar o modo de depuração para o Trident passando o `-d` sinalizador para o parâmetro de instalação com base na opção de instalação.

  Em seguida, confirme se a depuração é definida usando `./tridentctl logs -n trident` e pesquisando `level=debug msg` no log.

  Instalado com Operador

  ```
  kubectl patch torc trident -n <namespace> --type=merge -p
  '{"spec":{"debug":true}}'
  ```

  Isso reiniciará todos os pods do Trident, o que pode levar vários segundos. Você pode verificar isso observando a coluna 'IDADE' na saída do `kubectl get pod -n trident`.

  Para Astra Trident 20,07 e 20,10 utilização `tprov` em vez de `torc` .

  Instalado com Helm

  ```
  helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
  tridentDebug=true`
  ```

  Instalado com tridentctl

  ```
  ./tridentctl uninstall -n trident
  ./tridentctl install -d -n trident
  ```

- Você também pode obter logs de depuração para cada back-end, incluindo `debugTraceFlags` na sua definição de back-end. Por exemplo, inclua `debugTraceFlags: {“api”:true, “method”:true,}` para obter chamadas de API e transversais de método nos logs do Trident. Os backends existentes podem ter `debugTraceFlags` sido configurados com um `tridentctl backend update`.
- Ao usar o RedHat CoreOS, certifique-se de que `iscsid` está habilitado nos nós de trabalho e iniciado por padrão. Isso pode ser feito usando OpenShift MachineConfigs ou modificando os modelos de ignição.
- Um problema comum que você pode encontrar ao usar o Trident com "Azure NetApp Files" é quando os segredos do locatário e do cliente vêm de um Registro de aplicativo com permissões insuficientes. Para obter uma lista completa dos requisitos do Trident, consulte a "Azure NetApp Files" configuração.
- Se houver problemas com a montagem de um PV em um recipiente, certifique-se de que `rpcbind` está instalado e funcionando. Use o gerenciador de pacotes necessário para o sistema operacional do host e verifique se `rpcbind` está em execução. Você pode verificar o status `rpcbind` do serviço executando um `systemctl status rpcbind` ou seu equivalente.

- Se um back-end do Trident relatar que ele está `failed` no estado apesar de ter trabalhado antes, provavelmente será causado pela alteração das credenciais do SVM/administrador associadas ao back-end. Atualizar as informações de back-end usando `tridentctl update backend` ou saltando o pod Trident corrigirá esse problema.
- Se você encontrar problemas de permissão ao instalar o Trident com Docker como o runtime do contentor, tente a instalação do Trident com o `--in cluster=false` sinalizador. Isso não usará um pod do instalador e evitará problemas de permissão vistos devido ao `trident-installer` usuário.
- Utilize o `uninstall parameter <Uninstalling Trident>` para limpar após uma falha de funcionamento. Por padrão, o script não remove os CRDs que foram criados pelo Trident, tornando seguro desinstalar e instalar novamente mesmo em uma implantação em execução.
- Se você quiser fazer o downgrade para uma versão anterior do Trident, primeiro execute o `tridentctl uninstall` comando para remover o Trident. Baixe o desejado "Versão Trident" e instale usando o `tridentctl install` comando.
- Após uma instalação bem-sucedida, se um PVC estiver preso na `Pending` fase, a execução `kubectl describe pvc` pode fornecer informações adicionais sobre por que a Trident não conseguiu fornecer um PV para este PVC.

Implantação sem êxito do Trident usando o operador

Se você estiver implantando o Trident usando o operador, o status das `TridentOrchestrator` alterações será alterado de `Installing` para `Installed`. Se você observar o `Failed` status e o operador não conseguir se recuperar sozinho, você deve verificar os logs do operador executando o seguinte comando:

```
tridentctl logs -l trident-operator
```

Arrastar os logs do contentor do operador Trident pode apontar para onde está o problema. Por exemplo, um desses problemas poderia ser a incapacidade de extrair as imagens de contentor necessárias de Registros upstream em um ambiente com airgapped.

Para entender por que a instalação do Trident não foi bem-sucedida, você deve dar uma olhada no `TridentOrchestrator` status.

```
kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                          Message
  ----     ------  ----               ----                          -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

Este erro indica que já existe um `TridentOrchestrator` que foi usado para instalar o Trident. Como cada cluster do Kubernetes pode ter apenas uma instância do Trident, o operador garante que, a qualquer momento, só exista uma ativa `TridentOrchestrator` que possa criar.

Além disso, observar o status dos pods do Trident geralmente pode indicar se algo não está certo.

```
kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

Você pode ver claramente que os pods não são capazes de inicializar completamente porque uma ou mais imagens de contentor não foram obtidas.

Para resolver o problema, você deve editar o `TridentOrchestrator` CR. Alternativamente, você pode excluir `TridentOrchestrator` e criar um novo com a definição modificada e precisa.

Implantação sem êxito do Trident usando `tridentctl`

Para ajudar a descobrir o que deu errado, você pode executar o instalador novamente usando o `-d` argumento, que irá ativar o modo de depuração e ajudá-lo a entender qual é o problema:

```
./tridentctl install -n trident -d
```

Depois de resolver o problema, você pode limpar a instalação da seguinte forma e, em seguida, executar o `tridentctl install` comando novamente:

```
./tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Removed Trident user from security context constraint.
INFO Trident uninstallation succeeded.
```

Remova completamente o Astra Trident e CRDs

Você pode remover completamente o Astra Trident e todas as CRDs criadas e os recursos personalizados associados.

Isso não pode ser desfeito. Não faça isso a menos que você queira uma instalação completamente nova do Astra Trident. Para desinstalar o Astra Trident sem remover CRDs, "Desinstale o Astra Trident"consulte .

Operador Trident

Para desinstalar o Astra Trident e remover completamente CRDs usando o operador Trident:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

Leme

Para desinstalar o Astra Trident e remover completamente CRDs usando Helm:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

<code>dtridentctl</code>

Para remover completamente CRDs após desinstalar o Astra Trident usando `tridentctl`

```
tridentctl obliviate crd
```

Falha de desinstalação do nó NVMe com namespaces de bloco bruto RWX do Kubernetes 1,26

Se você estiver executando o Kubernetes 1,26, a desinstalação de nós pode falhar ao usar NVMe/TCP com namespaces de bloco bruto RWX. Os cenários a seguir fornecem uma solução para a falha. Como alternativa, você pode atualizar o Kubernetes para 1,27.

Excluiu o namespace e o pod

Considere um cenário em que você tenha um namespace gerenciado Astra Trident (volume persistente NVMe) anexado a um pod. Se você excluir o namespace diretamente do back-end do ONTAP, o processo de despreparo fica preso após tentar excluir o pod. Esse cenário não afeta o cluster do Kubernetes ou outras funcionalidades.

Solução alternativa

Desmonte o volume persistente (correspondente a esse namespace) do respetivo nó e exclua-o.

Dados bloqueados

```
 If you block (or bring down) all the dataLIFs of the NVMe Astra Trident
backend, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Solução alternativa
Abra o dataLIFS para restaurar a funcionalidade completa.
```

Mapeamento de namespace excluído

```
 If you remove the `hostNQN` of the worker node from the corresponding
subsystem, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Solução alternativa
Adicione o `hostNQN` de volta ao subsistema.
```

Suporte

O NetApp é compatível com o Astra Trident de várias maneiras. Amplas opções gratuitas de suporte autônomo estão disponíveis 24 horas por dia, 7 dias por semana, como artigos da base de conhecimento (KB) e um canal discord.

Ciclo de vida do suporte ao Astra Trident

O Astra Trident oferece três níveis de suporte com base na sua versão. "Suporte à versão do software NetApp para definições"Consulte a .

Suporte completo
: O Astra Trident fornece suporte total por doze meses a partir da data de lançamento.

Suporte limitado
: O Astra Trident fornece suporte limitado por meses, 13 a 24, a partir da data de lançamento.

Auto-suporte
: A documentação do Astra Trident está disponível durante os meses 25 a 36 a partir da data de lançamento.

Versão	Suporte completo	Suporte limitado	Auto-suporte
"24,02"	Fevereiro de 2025	Fevereiro de 2026	Fevereiro de 2027
"23,10"	Outubro de 2024	Outubro de 2025	Outubro de 2026
"23,07"	Julho de 2024	Julho de 2025	Julho de 2026

Versão	Suporte completo	Suporte limitado	Auto-suporte
"23,04"	Abril de 2024	Abril de 2025	Abril de 2026
"23,01"	—	Janeiro de 2025	Janeiro de 2026
"22,10"	—	Outubro de 2024	Outubro de 2025
"22,07"	—	Julho de 2024	Julho de 2025
"22,04"	—	Abril de 2024	Abril de 2025
"22,01"	—	—	Janeiro de 2025

Auto-suporte

Para obter uma lista abrangente de artigos de solução de problemas, "Base de Conhecimento NetApp (login necessário)" consulte . Você também encontrará informações sobre a solução de problemas relacionados ao Astra "aqui".

Apoio comunitário

Há uma vibrante comunidade pública de usuários de contêineres (incluindo desenvolvedores do Astra Trident"Canal discord") no nosso Astra . Este é um ótimo lugar para fazer perguntas gerais sobre o projeto e discutir tópicos relacionados com colegas de mentalidade semelhante.

Suporte técnico da NetApp

Para obter ajuda com o Astra Trident, crie um pacote de suporte usando `tridentctl logs -a -n trident` e envie para `NetApp Support <Getting Help>`o .

Para mais informações

- "Blogs do Astra"
- "Blogs do Astra Trident"
- "Kubernetes Hub"
- "NetApp.io"

Referência

Portas Astra Trident

Saiba mais sobre as portas que o Astra Trident usa para comunicação.

Portas Astra Trident

O Astra Trident se comunica pelas seguintes portas:

Porta	Finalidade
8443	Backchannel HTTPS
8001	Endpoint de métricas Prometheus
8000	SERVIDOR REST do Trident
17546	Porta de sonda de disponibilidade/disponibilidade usada pelos pods daemonset Trident

A porta da sonda de disponibilidade/disponibilidade pode ser alterada durante a instalação utilizando o `--probe-port` sinalizador. É importante garantir que essa porta não esteja sendo usada por outro processo nos nós de trabalho.

API REST do Astra Trident

"comandos e opções tridentctl"Embora seja a maneira mais fácil de interagir com a API REST do Astra Trident, você pode usar o endpoint REST diretamente, se preferir.

Quando usar a API REST

A API REST é útil para instalações avançadas que usam o Astra Trident como um binário autônomo em implantações não Kubernetes.

Para uma melhor segurança, o Astra Trident `REST API` é restrito a localhost por padrão ao ser executado dentro de um pod. Para alterar esse comportamento, você precisa definir o argumento do Astra Trident `-address` na configuração do pod.

Usando a API REST

Para exemplos de como essas APIs são chamadas, passe o (`-d`sinalizador debug). Para obter mais informações, "Gerenciar o Astra Trident usando o tridentctl"consulte .

A API funciona da seguinte forma:

OBTER

`GET <trident-address>/trident/v1/<object-type>`

Lista todos os objetos desse tipo.

`GET <trident-address>/trident/v1/<object-type>/<object-name>`

Obtém os detalhes do objeto nomeado.

POST

`POST <trident-address>/trident/v1/<object-type>`

Cria um objeto do tipo especificado.

- Requer uma configuração JSON para o objeto a ser criado. Para obter a especificação de cada tipo de objeto, "Gerenciar o Astra Trident usando o tridentctl"consulte a .
- Se o objeto já existir, o comportamento varia: Os backends atualizam o objeto existente, enquanto todos os outros tipos de objeto falharão a operação.

ELIMINAR

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`

Exclui o recurso nomeado.

> Os volumes associados a backends ou classes de armazenamento continuarão a existir; estes devem ser excluídos separadamente. Para obter mais informações, "Gerenciar o Astra Trident usando o tridentctl"consulte .

Opções de linha de comando

O Astra Trident expõe várias opções de linha de comando para o orquestrador Trident. Você pode usar essas opções para modificar sua implantação.

A registar

`-debug`

Ativa a saída de depuração.

`-loglevel <level>`

Define o nível de log (debug, info, warn, error, fatal). O padrão é INFO.

Kubernetes

`-k8s_pod`

Use essa opção ou `-k8s_api_server` para ativar o suporte do Kubernetes. Isso faz com que o Trident use suas credenciais de conta de serviço do Kubernetes do pod que contêm para entrar em Contato com o servidor de API. Isso só funciona quando o Trident é executado como um pod em um cluster do Kubernetes com contas de serviço ativadas.

`-k8s_api_server <insecure-address:insecure-port>`

Use essa opção ou `-k8s_pod` para ativar o suporte do Kubernetes. Quando especificado, o Trident se coneta ao servidor de API do Kubernetes usando o endereço e a porta inseguros fornecidos. Isso permite que o Trident seja implantado fora de um pod; no entanto, ele só suporta conexões inseguras com o servidor de API. Para se conetar com segurança, implante o Trident em um pod com a `-k8s_pod` opção.

Docker

`-volume_driver <name>`

Nome do driver usado ao Registrar o plug-in do Docker. O padrão é `netapp`.

`-driver_port <port-number>`

Ouça nesta porta em vez de um soquete de domínio UNIX.

`-config <file>`

Obrigatório; você deve especificar esse caminho para um arquivo de configuração de back-end.

DESCANSO

`-address <ip-or-host>`

Especifica o endereço no qual o servidor REST do Trident deve ouvir. O padrão é localhost. Ao ouvir no localhost e executar dentro de um pod Kubernetes, a interface REST não é diretamente acessível de fora do pod. `-address ""`Utilize para tornar a INTERFACE REST acessível a partir do endereço IP do pod.

> A interface REST DO Trident pode ser configurada para ouvir e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

`-port <port-number>`

Especifica a porta na qual o servidor REST do Trident deve ouvir. O padrão é 8000.

`-rest`

Ativa a interface REST. O padrão é verdadeiro.

Objetos Kubernetes e Trident

É possível interagir com o Kubernetes e o Trident usando APIs REST lendo e escrevendo objetos de recursos. Há vários objetos de recursos que ditam a relação entre o Kubernetes e o Trident, o Trident e o storage, o Kubernetes e o storage. Alguns desses objetos são gerenciados pelo Kubernetes e os outros são gerenciados pelo Trident.

Como os objetos interagem uns com os outros?

Talvez a maneira mais fácil de entender os objetos, para que eles são e como eles interagem seja seguir uma única solicitação de armazenamento de um usuário do Kubernetes:

1. Um usuário cria `PersistentVolumeClaim` uma solicitação de um novo `PersistentVolume` de um tamanho específico de um Kubernetes `StorageClass` que foi configurado anteriormente pelo administrador.
2. O Kubernetes `StorageClass` identifica o Trident como seu provisionador e inclui parâmetros que informam ao Trident como provisionar um volume para a classe solicitada.
3. O Trident olha para si `StorageClass` mesmo com o mesmo nome que identifica a correspondência `Backends` e `StoragePools` que pode usar para provisionar volumes para a classe.
4. O Trident provisiona o storage em um back-end compatível e cria dois objetos: Um `PersistentVolume` no Kubernetes que diz ao Kubernetes como encontrar, montar e tratar o volume e um volume no Trident

que mantém a relação entre o `PersistentVolume` e o storage real.

5. O Kubernetes vincula `PersistentVolumeClaim` o ao novo `PersistentVolume`. Pods que incluem a `PersistentVolumeClaim` montagem que Persistentvolume em qualquer host em que ele seja executado.
6. Um usuário cria um `VolumeSnapshot` de um PVC existente, usando um `VolumeSnapshotClass` que aponta para Trident.
7. O Trident identifica o volume que está associado ao PVC e cria um snapshot do volume em seu back-end. Ele também cria um `VolumeSnapshotContent` que instrui o Kubernetes sobre como identificar o snapshot.
8. Um usuário pode criar um `PersistentVolumeClaim` usando `VolumeSnapshot` como fonte.
9. O Trident identifica o instantâneo necessário e executa o mesmo conjunto de etapas envolvidas na criação de um `PersistentVolume` e um `Volume`.

Para ler mais sobre objetos do Kubernetes, é altamente recomendável que você leia a "Volumes persistentes" seção da documentação do Kubernetes.

Objetos do Kubernetes `PersistentVolumeClaim`

Um objeto Kubernetes `PersistentVolumeClaim` é uma solicitação de storage feita por um usuário de cluster do Kubernetes.

Além da especificação padrão, o Trident permite que os usuários especifiquem as seguintes anotações específicas de volume se quiserem substituir os padrões definidos na configuração de back-end:

Anotação	Opção de volume	Drivers suportados
Trident.NetApp.io/sistema de arquivos	Sistema de ficheiros	ONTAP-san, SolidFire-san, ONTAP-san-economy
Trident.NetApp.io/cloneFromPVC	CloneSourcevolume	ONTAP-nas, ONTAP-san, SolidFire-san, azure-NetApp-files, gcp-cvs, ONTAP-san-economy
Trident.NetApp.io/splitOnClone	SplitOnClone	ONTAP-nas, ONTAP-san
Trident.NetApp.io/protocolo	protocolo	qualquer
Trident.NetApp.io/exportPolicy	Política de exportação	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup
Trident.NetApp.io/snapshotPolicy	Política de SnapshotPolicy	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san
Trident.NetApp.io/snapshotServe	SnapshotServe	ONTAP-nas, ONTAP-nas-FlexGroup, ONTAP-san, gcp-cvs
Trident.NetApp.io/snapshotDirectory	SnapshotDirectory	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup
Trident.NetApp.io/unixPermissions	UnixPermissions	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup
Trident.NetApp.io/blocksize	Tamanho do bloco	SolidFire-san

Se o PV criado tiver a `Delete` política de recuperação, o Trident excluirá o PV e o volume de backup quando o PV for liberado (ou seja, quando o usuário exclui o PVC). Caso a ação de exclusão falhe, o Trident marca o PV como tal e periodicamente tenta novamente a operação até que seja bem-sucedida ou o PV seja excluído manualmente. Se o PV usar a `Retain` política, o Trident a ignora e assume que o administrador irá limpá-la do Kubernetes e do back-end, permitindo que o volume seja feito backup ou inspecionado antes de sua remoção. Observe que a exclusão do PV não faz com que o Trident exclua o volume de backup. Você deve removê-lo usando a API REST (`tridentctl`).

O Trident dá suporte à criação de snapshots de volume usando a especificação CSI: Você pode criar um instantâneo de volume e usá-lo como fonte de dados para clonar PVCs existentes. Dessa forma, cópias pontuais de PVS podem ser expostas ao Kubernetes na forma de snapshots. Os instantâneos podem então ser usados para criar novos PVS. Dê uma olhada `On-Demand Volume Snapshots` para ver como isso funcionaria.

O Trident também fornece as `cloneFromPVC` anotações e `splitOnClone` para a criação de clones. Você pode usar essas anotações para clonar um PVC sem precisar usar a implementação do CSI.

Aqui está um exemplo: Se um usuário já tem um PVC chamado `mysql`, o usuário pode criar um novo PVC chamado `mysqlclone` usando a anotação, como `trident.netapp.io/cloneFromPVC: mysql` . Com esse conjunto de anotações, o Trident clona o volume correspondente ao PVC mysql, em vez de provisionar um volume do zero.

Considere os seguintes pontos:

- Recomendamos clonar um volume ocioso.
- O PVC e seu clone devem estar no mesmo namespace do Kubernetes e ter a mesma classe de storage.
- Com os `ontap-nas` drivers e `ontap-san`, pode ser desejável definir a anotação de PVC `trident.netapp.io/splitOnClone` em conjunto `trident.netapp.io/cloneFromPVC` com o . Com `trident.netapp.io/splitOnClone` definido como `true`, o Trident divide o volume clonado do volume pai e, portanto, desacoplando completamente o ciclo de vida do volume clonado de seus pais às custas de perder alguma eficiência de storage. Não `trident.netapp.io/splitOnClone` configurá-lo ou configurá-lo para `false` resultar em consumo de espaço reduzido no back-end à custa de criar dependências entre os volumes pai e clone, de modo que o volume pai não possa ser excluído a menos que o clone seja excluído primeiro. Um cenário em que dividir o clone faz sentido é clonar um volume de banco de dados vazio, onde é esperado que o volume e seu clone diverjam muito e não se beneficiem das eficiências de armazenamento oferecidas pelo ONTAP.

O `sample-input` diretório contém exemplos de definições de PVC para uso com Trident. Consulte a para obter uma descrição completa dos parâmetros e definições associados aos volumes Trident.

Objetos do Kubernetes `PersistentVolume`

Um objeto Kubernetes `PersistentVolume` representa um storage disponibilizado para o cluster do Kubernetes. Ele tem um ciclo de vida que é independente do pod que o usa.

> O Trident cria `PersistentVolume` objetos e os Registra no cluster do Kubernetes automaticamente com base nos volumes provisionados. Você não é esperado para gerenciá-los sozinho.

Quando você cria um PVC que se refere a um Trident-based `StorageClass`, o Trident provisiona um novo volume usando a classe de armazenamento correspondente e Registra um novo PV para esse volume. Ao configurar o volume provisionado e o PV correspondente, o Trident segue as seguintes regras:

- O Trident gera um nome PV para o Kubernetes e um nome interno que ele usa para provisionar o storage. Em ambos os casos, é garantir que os nomes são únicos em seu escopo.
- O tamanho do volume corresponde ao tamanho solicitado no PVC o mais próximo possível, embora possa ser arredondado para a quantidade alocável mais próxima, dependendo da plataforma.

Objetos do Kubernetes `StorageClass`

Os objetos Kubernetes `StorageClass` são especificados por nome em `PersistentVolumeClaims` para provisionar o storage com um conjunto de propriedades. A própria classe de storage identifica o provisionador a ser usado e define esse conjunto de propriedades em termos que o provisionador entende.

É um dos dois objetos básicos que precisam ser criados e gerenciados pelo administrador. O outro é o objeto backend do Trident.

Um objeto do Kubernetes `StorageClass` que usa o Trident é parecido com este:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Esses parâmetros são específicos do Trident e informam à Trident como provisionar volumes para a classe.

Os parâmetros da classe de armazenamento são:

Atributo	Tipo	Obrigatório	Descrição
atributos	map[string]string	não	Veja a seção atributos abaixo
StoragePools	MAP[string]StringList	não	Mapa de nomes de back-end para listas de pools de armazenamento dentro
Além disso, StoragePools	MAP[string]StringList	não	Mapa de nomes de back-end para listas de pools de armazenamento dentro
Excluir StoragePools	MAP[string]StringList	não	Mapa de nomes de back-end para listas de pools de armazenamento dentro

Os atributos de storage e seus possíveis valores podem ser classificados em atributos de seleção de pool de storage e atributos do Kubernetes.

Atributos de seleção do pool de armazenamento

Esses parâmetros determinam quais pools de storage gerenciado pelo Trident devem ser utilizados para provisionar volumes de um determinado tipo.

Atributo	Tipo	Valores	Oferta	Pedido	Suportado por
1	cadeia de carateres	hdd, híbrido, ssd	Pool contém Mídia desse tipo; híbrido significa ambos	Tipo de material especificado	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, SolidFire-san
ProvisioningType	cadeia de carateres	fino, grosso	O pool é compatível com esse método de provisionamento	Método de provisionamento especificado	thick: all ONTAP; thin: all ONTAP & SolidFire-san
BackendType	cadeia de carateres	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, SolidFire-san, gcp-cvs, azure-NetApp-files, ONTAP-san-economy	Pool pertence a este tipo de backend	Back-end especificado	Todos os drivers
instantâneos	bool	verdadeiro, falso	O pool é compatível com volumes com snapshots	Volume com instantâneos ativados	ONTAP-nas, ONTAP-san, SolidFire-san, gcp-cvs
clones	bool	verdadeiro, falso	O pool é compatível com volumes de clonagem	Volume com clones ativados	ONTAP-nas, ONTAP-san, SolidFire-san, gcp-cvs
criptografia	bool	verdadeiro, falso	O pool é compatível com volumes criptografados	Volume com encriptação ativada	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-flexgroups, ONTAP-san
IOPS	int	número inteiro positivo	O pool é capaz de garantir IOPS nessa faixa	Volume garantido estas operações de entrada/saída por segundo	SolidFire-san

1: Não suportado pelos sistemas ONTAP Select

Na maioria dos casos, os valores solicitados influenciam diretamente o provisionamento; por exemplo, a solicitação de provisionamento espesso resulta em um volume provisionado rapidamente. No entanto, um pool de storage de elemento usa o mínimo e o máximo de IOPS oferecidos para definir valores de QoS, em vez do valor solicitado. Nesse caso, o valor solicitado é usado apenas para selecionar o pool de armazenamento.

O ideal é usar `attributes` sozinho para modelar as qualidades do storage de que você precisa para atender às necessidades de uma classe específica. O Trident deteta e seleciona automaticamente pools de armazenamento que correspondem a *all* do `attributes` que você especificar.

Se você não conseguir usar `attributes` para selecionar automaticamente os pools certos para uma classe, use os `storagePools` parâmetros e `additionalStoragePools` para refinar ainda mais os pools ou até mesmo selecionar um conjunto específico de pools.

Você pode usar o `storagePools` parâmetro para restringir ainda mais o conjunto de pools que correspondem a qualquer `attributes` especificado . Em outras palavras, o Trident usa a interseção de pools identificados pelos `attributes` parâmetros e `storagePools` para o provisionamento. Você pode usar um parâmetro sozinho ou ambos juntos.

Você pode usar o `additionalStoragePools` parâmetro para estender o conjunto de pools que o Trident usa para provisionamento, independentemente de quaisquer pools selecionados pelos `attributes` parâmetros e. `storagePools`

Você pode usar o `excludeStoragePools` parâmetro para filtrar o conjunto de pools que o Trident usa para provisionar. O uso desse parâmetro remove todos os pools que correspondem.

```
`storagePools`Nos parâmetros e `additionalStoragePools`, cada entrada assume o formulário `<backend>:<storagePoolList>`, onde `<storagePoolList>` é uma lista separada por vírgulas de pools de armazenamento para o back-end especificado. Por exemplo, um valor para `additionalStoragePools` pode parecer como `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Essas listas aceitam valores de regex tanto para os valores de backend quanto de lista. Você pode usar `tridentctl get backend` para obter a lista de backends e suas piscinas.
```

Atributos do Kubernetes

Esses atributos não têm impacto na seleção de pools de storage/back-ends pelo Trident durante o provisionamento dinâmico. Em vez disso, esses atributos simplesmente fornecem parâmetros compatíveis com volumes persistentes do Kubernetes. Os nós de trabalho são responsáveis pelas operações de criação de sistema de arquivos e podem exigir utilitários de sistema de arquivos, como xfsprogs.

Atributo	Tipo	Valores	Descrição	Drivers relevantes	Versão do Kubernetes
FsType	cadeia de carateres	ext4, ext3, xfs, etc.	O tipo de sistema de arquivos para volumes de bloco	SolidFire-san, ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, ONTAP-san-economy	Tudo
AllowVolumeExpansion	booleano	verdadeiro, falso	Ative ou desative o suporte para aumentar o tamanho do PVC	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, ONTAP-san-economy, SolidFire-san, gcp-cvs, azure-NetApp-files	Mais de 1,11 anos
VolumeBindingMode	cadeia de carateres	Imediato, WaitForFirstConsumer	Escolha quando ocorre a vinculação de volume e o provisionamento dinâmico	Tudo	1,19 - 1,26

- O `fsType` parâmetro é usado para controlar o tipo de sistema de arquivos desejado para LUNs SAN. Além disso, o Kubernetes também usa a presença de `fsType` em uma classe de armazenamento para indicar que existe um sistema de arquivos. A propriedade do volume só pode ser controlada usando o `fsGroup` contexto de segurança de um pod se `fsType` estiver definido. "Kubernetes: Configurar um contexto de segurança para um pod ou contêiner"Consulte para obter uma visão geral sobre como definir a propriedade do volume usando o `fsGroup` contexto. O Kubernetes aplicará o `fsGroup` valor somente se:
  - `fsType` é definido na classe de armazenamento.
  - O modo de acesso de PVC é RWO.

  Para drivers de armazenamento NFS, já existe um sistema de arquivos como parte da exportação NFS. Para usar `fsGroup` a classe de armazenamento ainda precisa especificar um `fsType`. você pode configurá-lo como `nfs` ou qualquer valor não nulo.
- "Expanda volumes"Consulte para obter mais detalhes sobre a expansão do volume.
- O pacote de instalação do Trident fornece vários exemplos de definições de classe de armazenamento para uso com o Trident no `sample-input/storage-class-*.yaml`. A exclusão de uma classe de armazenamento Kubernetes faz com que a classe de armazenamento Trident correspondente também seja excluída.

Objetos do Kubernetes `VolumeSnapshotClass`

Os objetos do Kubernetes `VolumeSnapshotClass` são análogos ao `StorageClasses`. Eles ajudam a definir várias classes de armazenamento e são referenciados por instantâneos de volume para associar o snapshot à classe de snapshot necessária. Cada snapshot de volume é associado a uma classe de snapshot de volume único.

A `VolumeSnapshotClass` deve ser definida por um administrador para criar instantâneos. Uma classe de instantâneo de volume é criada com a seguinte definição:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

O `driver` especifica ao Kubernetes que as solicitações de snapshots de volume `csi-snapclass` da classe são tratadas pelo Trident. O `deletionPolicy` especifica a ação a ser tomada quando um instantâneo deve ser excluído. `deletionPolicy`Quando está definido como `Delete`, os objetos instantâneos de volume e o instantâneo subjacente no cluster de armazenamento são removidos quando um instantâneo é excluído. Alternativamente, configurá-lo para `Retain` significa que `VolumeSnapshotContent` e o instantâneo físico são retidos.

Objetos do Kubernetes `VolumeSnapshot`

Um objeto Kubernetes `VolumeSnapshot` é uma solicitação para criar um snapshot de um volume. Assim como um PVC representa uma solicitação feita por um usuário para um volume, um instantâneo de volume é uma solicitação feita por um usuário para criar um instantâneo de um PVC existente.

Quando uma solicitação de snapshot de volume entra, o Trident gerencia automaticamente a criação do snapshot para o volume no back-end e expõe o snapshot criando um objeto exclusivo `VolumeSnapshotContent`. Você pode criar snapshots a partir de PVCs existentes e usar os snapshots como DataSource ao criar novos PVCs.

> A vida útil de um VolumeSnapshot é independente do PVC de origem: Um snapshot persiste mesmo depois que o PVC de origem é excluído. Ao excluir um PVC que tenha instantâneos associados, o Trident marca o volume de apoio para este PVC em um estado **Deletando**, mas não o remove completamente. O volume é removido quando todos os instantâneos associados são excluídos.

Objetos do Kubernetes `VolumeSnapshotContent`

Um objeto Kubernetes `VolumeSnapshotContent` representa um snapshot retirado de um volume já provisionado. Ele é análogo a `PersistentVolume` e significa um snapshot provisionado no cluster de storage. Semelhante aos `PersistentVolumeClaim` objetos e `PersistentVolume`, quando um snapshot é criado, o `VolumeSnapshotContent` objeto mantém um mapeamento um-para-um para o `VolumeSnapshot` objeto, que havia solicitado a criação do snapshot.

O `VolumeSnapshotContent` objeto contém detalhes que identificam exclusivamente o instantâneo, como o

`snapshotHandle`. Esta `snapshotHandle` é uma combinação única do nome do PV e do nome do `VolumeSnapshotContent` objeto.

Quando uma solicitação de snapshot entra, o Trident cria o snapshot no back-end. Depois que o snapshot é criado, o Trident configura um `VolumeSnapshotContent` objeto e, portanto, expõe o snapshot à API do Kubernetes.

> Normalmente, você não precisa gerenciar o `VolumeSnapshotContent` objeto. Uma exceção a isso é quando você deseja "importar um instantâneo de volume"criar fora do Astra Trident.

Objetos do Kubernetes `CustomResourceDefinition`

Os recursos personalizados do Kubernetes são endpoints na API do Kubernetes que são definidos pelo administrador e são usados para agrupar objetos semelhantes. O Kubernetes dá suporte à criação de recursos personalizados para armazenar uma coleção de objetos. Você pode obter essas definições de recursos executando `kubectl get crds`o .

As definições personalizadas de recursos (CRDs) e os metadados de objetos associados são armazenados pelo Kubernetes em seu armazenamento de metadados. Isso elimina a necessidade de uma loja separada para o Trident.

O Astra Trident usa `CustomResourceDefinition` objetos para preservar a identidade de objetos Trident, como back-ends Trident, classes de storage Trident e volumes Trident. Esses objetos são gerenciados pelo Trident. Além disso, a estrutura de snapshot do volume CSI introduz algumas CRDs que são necessárias para definir snapshots de volume.

CRDs são uma construção do Kubernetes. Os objetos dos recursos definidos acima são criados pelo Trident. Como um exemplo simples, quando um back-end é criado usando `tridentctl`o , um objeto CRD correspondente `tridentbackends` é criado para consumo pelo Kubernetes.

Aqui estão alguns pontos a ter em mente sobre os CRDs do Trident:

- Quando o Trident é instalado, um conjunto de CRDs é criado e pode ser usado como qualquer outro tipo de recurso.
- Ao desinstalar o Trident usando o `tridentctl uninstall` comando, os pods Trident são excluídos, mas os CRDs criados não são limpos. "Desinstale o Trident"Consulte para compreender como o Trident pode ser completamente removido e reconfigurado do zero.

Objetos Astra Trident `StorageClass`

O Trident cria classes de storage correspondentes para objetos Kubernetes `StorageClass` que especificam `csi.trident.netapp.io` no campo do provisionador. O nome da classe de storage corresponde ao do objeto Kubernetes `StorageClass` que ele representa.

> Com o Kubernetes, esses objetos são criados automaticamente quando um Kubernetes `StorageClass` que usa o Trident como provisionador é registrado.

As classes de armazenamento compreendem um conjunto de requisitos para volumes. O Trident atende a esses requisitos com os atributos presentes em cada pool de storage. Se forem correspondentes, esse pool de storage será um destino válido para volumes de provisionamento que usam essa classe de storage.

Você pode criar configurações de classe de armazenamento para definir diretamente classes de

armazenamento usando a API REST. No entanto, para implantações do Kubernetes, esperamos que elas sejam criadas ao Registrar novos objetos do Kubernetes `StorageClass`.

Objetos de back-end do Astra Trident

Os backends representam os fornecedores de storage em cima dos quais o Trident provisiona volumes. Uma única instância do Trident pode gerenciar qualquer número de backends.

Este é um dos dois tipos de objetos que você cria e gerencia a si mesmo. O outro é o objeto Kubernetes `StorageClass`.

Para obter mais informações sobre como construir esses objetos, "configurando backends"consulte .

Objetos Astra Trident `StoragePool`

Os pools de storage representam locais distintos disponíveis para provisionamento em cada back-end. Para ONTAP, eles correspondem a agregados em SVMs. Para NetApp HCI/SolidFire, estes correspondem a bandas de QoS especificadas pelo administrador. Para o Cloud Volumes Service, eles correspondem a regiões de provedores de nuvem. Cada pool de storage tem um conjunto de atributos de storage distintos, que definem suas características de performance e proteção de dados.

Ao contrário dos outros objetos aqui, os candidatos ao pool de armazenamento são sempre descobertos e gerenciados automaticamente.

Objetos Astra Trident `Volume`

Os volumes são a unidade básica de provisionamento, incluindo pontos de extremidade de back-end, como compartilhamentos NFS e iSCSI LUNs. No Kubernetes, eles correspondem diretamente `PersistentVolumes` ao . Ao criar um volume, certifique-se de que ele tenha uma classe de armazenamento, que determina onde esse volume pode ser provisionado, juntamente com um tamanho.

- No Kubernetes, esses objetos são gerenciados automaticamente. Você pode visualizá-los para ver o que o Trident provisionou.
- Ao excluir um PV com instantâneos associados, o volume Trident correspondente é atualizado para um estado **Deletando**. Para que o volume Trident seja excluído, você deve remover os snapshots do volume.

Uma configuração de volume define as propriedades que um volume provisionado deve ter.

Atributo	Tipo	Obrigatório	Descrição
versão	cadeia de carateres	não	Versão da API Trident ("1")
nome	cadeia de carateres	sim	Nome do volume a criar
StorageClass	cadeia de carateres	sim	Classe de storage a ser usada ao provisionar o volume
tamanho	cadeia de carateres	sim	Tamanho do volume a provisionar em bytes

Atributo	Tipo	Obrigatório	Descrição
protocolo	cadeia de carateres	não	Tipo de protocolo a utilizar; "ficheiro" ou "bloco"
InternalName	cadeia de carateres	não	Nome do objeto no sistema de storage; gerado pelo Trident
CloneSourcevolume	cadeia de carateres	não	ONTAP (nas, san) & SolidFire-*: Nome do volume a partir do qual clonar
SplitOnClone	cadeia de carateres	não	ONTAP (nas, san): Divida o clone de seu pai
Política de SnapshotPolicy	cadeia de carateres	não	ONTAP-*: Política de snapshot a ser usada
SnapshotServe	cadeia de carateres	não	ONTAP-*: Porcentagem de volume reservado para snapshots
Política de exportação	cadeia de carateres	não	ONTAP-nas*: Política de exportação para usar
SnapshotDirectory	bool	não	ONTAP-nas*: Se o diretório snapshot está visível
UnixPermissions	cadeia de carateres	não	ONTAP-nas*: Permissões iniciais do UNIX
Tamanho do bloco	cadeia de carateres	não	SolidFire-*: Tamanho do bloco/setor
Sistema de ficheiros	cadeia de carateres	não	Tipo de sistema de ficheiros

O Trident gera `internalName` ao criar o volume. Isto consiste em duas etapas. Primeiro, ele prepende o prefixo de armazenamento (o padrão `trident` ou o prefixo na configuração de back-end) para o nome do volume, resultando em um nome do formulário `<prefix>-<volume-name>`. Em seguida, procede à higienização do nome, substituindo carateres não permitidos no backend. Para backends ONTAP, ele substitui hífens por sublinhados (assim, o nome interno se torna `<prefix>_<volume-name>`). Para backends de elemento, ele substitui sublinhados por hífens.

Você pode usar configurações de volume para provisionar volumes diretamente usando a API REST, mas nas implantações do Kubernetes, esperamos que a maioria dos usuários use o método padrão do Kubernetes `PersistentVolumeClaim`. O Trident cria esse objeto de volume automaticamente como parte do processo de provisionamento.

Objetos Astra Trident `Snapshot`

Os snapshots são uma cópia pontual de volumes, que pode ser usada para provisionar novos volumes ou restaurar o estado. No Kubernetes, eles correspondem diretamente a `VolumeSnapshotContent` objetos. Cada snapshot é associado a um volume, que é a origem dos dados do snapshot.

Cada `Snapshot` objeto inclui as propriedades listadas abaixo:

Atributo	Tipo	Obrigatório	Descrição
versão	Cadeia de carateres	Sim	Versão da API Trident ("1")
nome	Cadeia de carateres	Sim	Nome do objeto snapshot Trident
InternalName	Cadeia de carateres	Sim	Nome do objeto snapshot Trident no sistema de storage
Nome do volume	Cadeia de carateres	Sim	Nome do volume persistente para o qual o instantâneo é criado
VolumeInternalName	Cadeia de carateres	Sim	Nome do objeto de volume Trident associado no sistema de storage

No Kubernetes, esses objetos são gerenciados automaticamente. Você pode visualizá-los para ver o que o Trident provisionou.

Quando uma solicitação de objeto Kubernetes `VolumeSnapshot` é criada, o Trident funciona criando um objeto snapshot no sistema de storage de backup. `internalName`O deste objeto instantâneo é gerado combinando o prefixo `snapshot-` com o `UID` do `VolumeSnapshot` objeto (por exemplo, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` e `volumeInternalName` são preenchidos obtendo os detalhes do volume de apoio.

Objeto Astra Trident `ResourceQuota`

O deamonset do Trident consome uma `system-node-critical` classe de prioridade - a classe de prioridade mais alta disponível no Kubernetes - para garantir que o Astra Trident possa identificar e limpar volumes durante o desligamento gracioso do nó e permitir que os pods do Trident daemonset pré-empt cargas de trabalho com prioridade mais baixa em clusters onde há alta pressão de recursos.

Para conseguir isso, o Astra Trident emprega um `ResourceQuota` objeto para garantir que uma classe de prioridade "crítica do nó do sistema" no daemonset do Trident esteja satisfeita. Antes da implantação e criação do daemonset, o Astra Trident procura o `ResourceQuota` objeto e, se não for descoberto, o aplica.

Se você precisar de mais controle sobre a cota de recurso padrão e Classe de prioridade, você pode gerar um `custom.yaml` ou configurar o `ResourceQuota` objeto usando o gráfico de Helm.

O seguinte é um exemplo de um objeto 'ResourceQuota' priorizando o daemonset do Trident.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
     matchExpressions:
       - operator : In
         scopeName: PriorityClass
         values: ["system-node-critical"]
```

Para obter mais informações sobre cotas de recursos, "Kubernetes: Cotas de recursos"consulte .

Limpe `ResourceQuota` se a instalação falhar

No caso raro em que a instalação falha depois que o `ResourceQuota` objeto é criado, primeiro "desinstalação"tente e depois reinstale.

Se isso não funcionar, remova manualmente o `ResourceQuota` objeto.

Retire `ResourceQuota`

Se você preferir controlar sua própria alocação de recursos, poderá remover o objeto Astra Trident `ResourceQuota` usando o comando:

```
kubectl delete quota trident-csi -n trident
```

Padrões de segurança do pod (PSS) e restrições de contexto de segurança (SCC)

Os padrões de segurança do pod do Kubernetes (PSS) e as políticas de segurança do Pod (PSP) definem níveis de permissão e restringem o comportamento dos pods. As restrições de contexto de Segurança OpenShift (SCC) definem similarmente a restrição de pod específica ao OpenShift Kubernetes Engine. Para oferecer essa personalização, o Astra Trident habilita certas permissões durante a instalação. As seções a seguir detalham as permissões definidas pelo Astra Trident.

O PSS substitui as políticas de segurança do Pod (PSP). A PSP foi obsoleta no Kubernetes v1,21 e será removida em v1,25. Para obter mais informações, "Kubernetes: Segurança"consulte .

Contexto de segurança do Kubernetes necessário e campos relacionados

Permissão	Descrição
Privilegiado	O CSI exige que os pontos de montagem sejam bidirecionais, o que significa que o pod de nó do Trident deve executar um contentor privilegiado. Para obter mais informações, "Kubernetes: Propagação de montagem"consulte .
Rede de host	Necessário para o daemon iSCSI. `iscsiadm` Gerencia montagens iSCSI e usa redes de host para se comunicar com o daemon iSCSI.
IPC do host	O NFS usa comunicação entre processos (IPC) para se comunicar com o NFSD.
PID do host	Necessário para iniciar `rpc-statd` o NFS. O Astra Trident consulta os processos de host para determinar se `rpc-statd` está em execução antes da montagem de volumes NFS.

Recursos	O `SYS_ADMIN` recurso é fornecido como parte dos recursos padrão para contentores privilegiados. Por exemplo, o Docker define esses recursos para contentores privilegiados:
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	O perfil Seccomp é sempre "unconfinado" em contentores privilegiados; portanto, não pode ser habilitado no Astra Trident.
SELinux	No OpenShift, os contentores privilegiados são executados no `spc_t` domínio ("contentor Super privilegiado") e os contentores sem privilégios são executados `container_t` no domínio. No `containerd`, com `container-selinux` instalado, todos os contentores são executados no `spc_t` domínio, o que desativa efetivamente o SELinux. Portanto, o Astra Trident não é adicionado `seLinuxOptions` a contêineres.
DAC	Os contentores privilegiados devem ser executados como root. Os contentores não privilegiados são executados como root para acessar os sockets unix exigidos pelo CSI.

Padrões de segurança do pod (PSS)

Etiqueta	Descrição	Padrão
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Permite que o controlador Trident e os nós sejam admitidos no namespace de instalação. Não altere a etiqueta do namespace.	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

> Alterar os rótulos do namespace pode resultar em pods não sendo programados, um "erro ao criar: …" ou, "Aviso: Trident-csi-…". Se isso acontecer, verifique se a etiqueta do namespace para `privileged` foi alterada. Em caso afirmativo, reinstale o Trident.

Políticas de segurança do pod (PSP)

Campo	Descrição	Padrão
`allowPrivilegeEscalation`	Os contêineres privilegiados devem permitir o escalonamento de privilégios.	`true`
`allowedCSIDrivers`	O Trident não usa volumes efêmeros de CSI inline.	Vazio
`allowedCapabilities`	Os contêineres Trident não privilegiados não exigem mais recursos do que o conjunto padrão e os contentores privilegiados recebem todos os recursos possíveis.	Vazio
`allowedFlexVolumes`	O Trident não faz uso de um "Controlador Flexvolume", portanto, eles não estão incluídos na lista de volumes permitidos.	Vazio
`allowedHostPaths`	O pod de nó Trident monta o sistema de arquivos raiz do nó, portanto, não há benefício para definir esta lista.	Vazio
`allowedProcMountTypes`	O Trident não usa nenhum `ProcMountTypes`.	Vazio
`allowedUnsafeSysctls`	O Trident não requer nenhum inseguro `sysctls`.	Vazio
`defaultAddCapabilities`	Não são necessários recursos para serem adicionados a contentores privilegiados.	Vazio
`defaultAllowPrivilegeEscalation`	Permitir o escalonamento de privilégios é Tratado em cada pod Trident.	`false`
`forbiddenSysctls`	Não `sysctls` são permitidos.	Vazio

Campo	Descrição	Padrão
`fsGroup`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`hostIPC`	A montagem de volumes NFS requer que o IPC do host se comunique com `nfsd`	`true`
`hostNetwork`	O iscsiadm requer que a rede host se comunique com o daemon iSCSI.	`true`
`hostPID`	O PID do host é necessário para verificar se `rpc-statd` está sendo executado no nó.	`true`
`hostPorts`	O Trident não usa nenhuma porta de host.	Vazio
`privileged`	Os pods de nós do Trident devem executar um contêiner privilegiado para montar volumes.	`true`
`readOnlyRootFilesystem`	Os pods de nós do Trident devem gravar no sistema de arquivos do nó.	`false`
`requiredDropCapabilities`	Os pods de nós do Trident executam um contêiner privilegiado e não podem descartar recursos.	`none`
`runAsGroup`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`runAsUser`	Os contêineres do Trident são executados como raiz.	`runAsAny`
`runtimeClass`	O Trident não usa `RuntimeClasses`o .	Vazio
`seLinux`	O Trident não define `seLinuxOptions` porque atualmente existem diferenças em como os tempos de execução de contêineres e as distribuições do Kubernetes lidam com o SELinux.	Vazio
`supplementalGroups`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`volumes`	Os pods do Trident exigem esses plugins de volume.	`hostPath, projected, emptyDir`

Restrições de contexto de segurança (SCC)

Etiquetas	Descrição	Padrão
`allowHostDirVolumePlugin`	Os pods de nó Trident montam o sistema de arquivos raiz do nó.	`true`
`allowHostIPC`	A montagem de volumes NFS requer que o IPC do host se comunique com `nfsd`o .	`true`
`allowHostNetwork`	O iscsiadm requer que a rede host se comunique com o daemon iSCSI.	`true`
`allowHostPID`	O PID do host é necessário para verificar se `rpc-statd` está sendo executado no nó.	`true`
`allowHostPorts`	O Trident não usa nenhuma porta de host.	`false`
`allowPrivilegeEscalation`	Os contêineres privilegiados devem permitir o escalonamento de privilégios.	`true`
`allowPrivilegedContainer`	Os pods de nós do Trident devem executar um contêiner privilegiado para montar volumes.	`true`
`allowedUnsafeSysctls`	O Trident não requer nenhum inseguro `sysctls`.	`none`
`allowedCapabilities`	Os contêineres Trident não privilegiados não exigem mais recursos do que o conjunto padrão e os contentores privilegiados recebem todos os recursos possíveis.	Vazio
`defaultAddCapabilities`	Não são necessários recursos para serem adicionados a contentores privilegiados.	Vazio
`fsGroup`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`groups`	Este SCC é específico do Trident e está vinculado ao seu usuário.	Vazio
`readOnlyRootFilesystem`	Os pods de nós do Trident devem gravar no sistema de arquivos do nó.	`false`
`requiredDropCapabilities`	Os pods de nós do Trident executam um contêiner privilegiado e não podem descartar recursos.	`none`
`runAsUser`	Os contêineres do Trident são executados como raiz.	`RunAsAny`

Etiquetas	Descrição	Padrão
`seLinuxContext`	O Trident não define `seLinuxOptions` porque atualmente existem diferenças em como os tempos de execução de contêineres e as distribuições do Kubernetes lidam com o SELinux.	Vazio
`seccompProfiles`	Os contentores privilegiados funcionam sempre "sem confinamentos".	Vazio
`supplementalGroups`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`users`	Uma entrada é fornecida para vincular esse SCC ao usuário Trident no namespace Trident.	n/a.
`volumes`	Os pods do Trident exigem esses plugins de volume.	`hostPath, downwardAPI, projected, emptyDir`

Avisos legais

Avisos legais fornecem acesso a declarações de direitos autorais, marcas registradas, patentes e muito mais.

Direitos de autor

"https://www.netapp.com/company/legal/copyright/"

Marcas comerciais

NetApp, o logotipo DA NetApp e as marcas listadas na página de marcas comerciais da NetApp são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respetivos proprietários.

"https://www.netapp.com/company/legal/trademarks/"

Patentes

Uma lista atual de patentes de propriedade da NetApp pode ser encontrada em:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Política de privacidade

"https://www.netapp.com/company/legal/privacy-policy/"

Código aberto

Você pode revisar os direitos autorais e as licenças de terceiros usadas no software NetApp para Astra Trident no arquivo de avisos de cada versão em https://github.com/NetApp/trident/.

Informações sobre direitos autorais

Informações sobre marcas comerciais